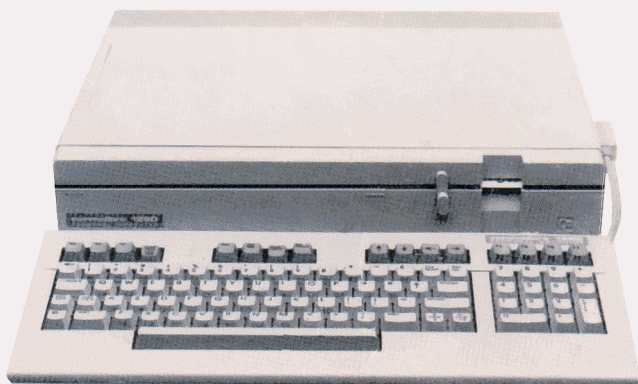
PERSONAL COMPUTER

Commodore 128D

PERSONAL COMPUTER

Manuale di Sistema

Personal Computer

Manuale di Sistema

Attenzione: Tutte le informazioni sul C 128 contenute in questo manuale, sono valide anche per il C 128 D

INDICE

MANUALE DI SISTEMA C 128

Capitolo I - Introduzione

Capitolo II - Uso della modalità C128

Capitolo III - Uso della modalità C64

Capitolo IV - Uso della modalità CP/M

Capitolo V - Enciclopedia Basic 7.0

Appendici

CAPITOLO 1

INTRODUZIONE

SEZIONE 1

Guida alla consultazione del manuale

Questo **Manuale di Sistema del Commodore 128** è stato concepito per permettere all'utente di sfruttare appieno le funzioni avanzate del computer Commodore 128. Di seguito viene spiegato come consultare il manuale.

Prima di continuare la lettura di questo **Manuale di Sistema**, si consiglia di leggere l'altro libro fornito con il computer, **Guida Introduttiva del Personal Computer Commodore 128**, che contiene informazioni importanti su come preparare ed avviare il Commodore 128.

Per chi è interessato principalmente all'uso del linguaggio BASIC per creare ed eseguire propri programmi, si consiglia di leggere per prima cosa il Capitolo II, USO DELLA MODALITÀ C128. Questo capitolo introduce il linguaggio di programmazione BASIC, utilizzato per le modalità C128 e C64; descrive la tastiera del Commodore 128; fornisce istruzioni su come utilizzare i comandi avanzati nelle modalità C128 e C64; spiega come utilizzare i nuovi numerosi e potenti comandi BASIC (compreso i comandi di colore, grafica e suono), da utilizzare unicamente in modalità C128. Questo capitolo descrive inoltre come utilizzare le funzioni a 80 colonne in modalità C128.

Per poter utilizzare il BASIC in modalità C64, leggere il Capitolo III, USO DELLA MODALITÀ C64.

Per utilizzare il CP/M con il Commodore 128, leggere il capitolo IV, USO DELLA MODALITÀ CP/M. Questo capitolo spiega come avviare ed usare il CP/M con il Commodore 128. In questa modalità si potranno utilizzare le migliaia di pacchetti software compresi nella serie PERFECT (PERFECT WRITER, PERFECT CALC, PERFECT FILER) e si potranno anche creare propri programmi in CP/M.

Per ulteriori dettagli sui comandi BASIC 7.0 consultare il Capitolo V, ENCICLOPEDIA BASIC 7.0. Questo capitolo fornisce informazioni sul formato e sull'utilizzo di tutti i comandi, istruzioni e funzioni del BASIC 7.0.

Se, dopo aver letto i Capitoli da I a V sono necessarie ulteriori informazioni tecniche su un argomento particolare a proposito del Commodore 128, consultare le Appendici in questo **Manuale di Sistema**. Le appendici contengono moltissime informazioni, come

ad esempio una lista dei messaggi di errore BASIC e DOS e un riassunto dei comandi disco. Subito dopo le Appendici si potrà trovare un Glossario contenente alcune definizioni di termini di informatica.

Per informazioni tecniche complete sulle caratteristiche del Commodore 128, consultare il Manuale di Riferimento per il Programmatore del Commodore 128.

SEZIONE 2

Informazioni generali sul Personal Computer Commodore C128

Informazioni generali sul Personal Computer Commodore 128

Il Personal Computer Commodore 128 offre tre principali modalità di funzionamento:

- Modalità C128
- Modalità C64
- Modalità CP/M

Ecco le caratteristiche di ogni modalità:

Modalità C128

In modalità C128 il Personal Computer Commodore 128 fornisce l'accesso a 128K di RAM ed un potente linguaggio BASIC esteso chiamato BASIC 7.0. Il BASIC 7.0 - che dispone di più di 140 comandi, istruzioni e funzioni. La modalità C128 offre inoltre la possibilità di lavorare su 40 o 80 colonne, con una tastiera a 92 tasti. Un monitor incorporato del linguaggio macchina permette di creare e di mettere a punto i propri programmi in linguaggio macchina. In modalità C128 si potranno utilizzare i molti e nuovi dispositivi periferici della Commodore, compreso un nuovo disk drive seriale veloce (mod. 1570), un mouse ed un monitor composito video/RGBI (mod. 1901) a 40/80 colonne, ed inoltre tutte le normali periferiche seriali della Commodore.

Modalità C64

In modalità C64 il Commodore 128 funziona esattamente come un computer Commodore 64, permettendo così l'uso dell'ampia gamma di programmi del Commodore 64. È inoltre totalmente compatibile con le periferiche del Commodore 64.

La modalità C64 offre linguaggio BASIC 2.0, output a 40 colonne ed accesso a 64K di RAM.

Modalità CP/M

In modalità CP/M, un microprocessore Z80 montato su piastra fornisce tutte le caratteristiche del CP/M della Digital Research, Versione 3.0, in aggiunta alle ulteriori capacità fornite dalla Commodore. Il pacchetto CP/M del Commodore 128, chiamato CP/M Plus, mette a disposizione 128K di memoria RAM, 40 e 80 colonne di output, accesso alla tastiera completa, al tastierino numerico ed ai tasti speciali, ed accesso al nuovo disk drive seriale veloce 1570 ed alle periferiche seriali standard.

I Capitoli II, III e IV, che contengono le sezioni da 3 a 15, danno informazioni sull'accesso e sull'uso delle capacità delle tre potenti e versatili modalità operative del Personal Computer Commodore 128.

PASSAGGIO DA UNA MODALITÀ ALL'ALTRA

La seguente tabella indica come passare da una modalità all'altra.

A \ DA	OFF	C128 40 col	C128 80 col	C64	CP/M 40 col	CP/M 80 col
C 128 40 col	1. Controllare che il tasto 40/80 sia sollevato. 2. Accendere il computer.		1. Premere il tasto ESC. 2. Premere il tasto X OPPURE 1. Controllare che il tasto 40/80 sia sollevato. 2. Premere il pulsante RESET.	1. Controllare che il tasto 40/80 sia sollevato. 2. Spegnere e riaccendere il computer.	1. Controllare che il tasto 40/80 sia sollevato. 2. Spegnere e riaccendere il computer.	1. Controllare che il tasto 40/80 sia sollevato. 2. Spegnere e riaccendere il computer.
C128 80 col	1. Premere il tasto 40/80. 2. Accendere il computer.	1. Premere il tasto ESC. 2. Premere il tasto X OPPURE 1. Premere il tasto 40/80. 2. Premere il pulsante RESET.		1. Premere il tasto 40/80. 2. Spegnere e riaccendere il computer.	1. Premere il tasto 40/80. 2. Togliere il disco di sistema CP/M dal drive, se necessario. 3. Spegnere e riaccendere il computer.	1. Premere il tasto 40/80. 2. Togliere il disco di sistema CP/M dal drive, se necessario. 3. Spegnere e riaccendere il computer.

C64	1. Tenere premuto il tasto C=. 2. Accendere il computer OPPURE 1. Introdurre la cartuccia C64. 2. Accendere il computer.	1. Battere GO 64; premere RETURN. 2. Appare il messaggio ARE YOU SURE? Battere Y e premere RETURN.	1. Battere GO 64; premere RETURN. 2. Appare il messaggio: ARE YOU SURE? Battere Y e premere RETURN.		1. Spegnere il computer. 2. Controllare che il tasto 40/80 sia sollevato. 3. Tenere premuto il tasto C= e accendere contemporaneamente il computer OPPURE 1. Spegnere il computer. 2. Introdurre la cartuccia C64. 3. Accendere il computer.	1. Spegnere il computer. 2. Controllare che il tasto 40/80 sia sollevato. 3. Tenere premuto il tasto C= e accendere contemporaneamente il computer OPPURE 1. Spegnere il computer. 2. Introdurre la cartuccia C64. 3. Accendere il computer.
CP/M 40 col	1. Accendere il disk drive. 2. Introdurre il disco di sistema CP/M nel drive. 3. Controllare che il tasto 40/80 sia sollevato. 4. Accendere il computer.	1. Accendere il disk drive. 2. Introdurre il disco di sistema CP/M nel drive. 3. Controllare che il tasto 40/80 sia sollevato. 4. Battere: BOOT. 5. Premere RETURN.	1. Accendere il disk drive. 2. Introdurre il disco di sistema CP/M nel drive. 3. Controllare che il tasto 40/80 sia sollevato. 4. Battere: BOOT 5. Premere RETURN	1. Controllare che il tasto 40/80 sia sollevato. 2. Accendere il disk drive. 3. Introdurre il disco di sistema CP/M nel drive. 4. Spegnere ed accendere il computer.		1. Introdurre il disco di utilità CP/M nel drive. 2. Al prompt: A>, battere: DEVICE CONOUT=40 col. 3. Premere RETURN.

CP/M 80 col	1. Accendere il disk drive. 2. Introdurre il disco di sistema CP/M nel drive. 3. Premere il tasto 40/80. 4. Accendere il computer.	1. Accendere il disk drive. 2. Introdurre il disco di sistema CP/M nel drive. 3. Premere il tasto 40/80. 4. Battere: BOOT. 5. Premere RETURN.	1. Accendere il disk drive. 2. Introdurre il disco di sistema CP/M nel drive. 3. Controllare che il tasto 40/80 sia sollevato. 4. Battere: BOOT 5. Premere RETURN	1. Premere il tasto 40/80. 2. Accendere il disk drive. 3. Introdurre il disco di sistema CP/M nel drive. 4. Spegnere ed accendere il computer.	1. Introdurre il disco di utilità CP/M nel drive. 2. Al prompt: A>, battere: DEVICE CONOUT=40 col. 3. Premere RETURN.

NOTA: Se si sta utilizzando un monitor duale Commodore 1901, ricordare di spostare l'interruttore del video da COMPOSITO o SEPARATO a RGBI quando si passa da 40 a 80 colonne; eseguire questa operazione al contrario quando si desidera passare da 80 a 40 colonne. Inoltre, per passare da una modalità all'altra, togliere l'eventuale cartuccia dalla porta di espansione o l'eventuale disco dal drive.

CAPITOLO 2

USO DELLA MODALITÀ C128

SEZIONE 3

Informazioni generali sulla programmazione in Basic

Linguaggio di programmazione BASIC

Il linguaggio di programmazione BASIC è un linguaggio speciale che permette di comunicare con il Commodore 128. Il BASIC è un mezzo utilizzato per comunicare al computer le azioni da intraprendere.

Il BASIC possiede un proprio vocabolario (composto da **comandi, istruzioni e funzioni**) e proprie regole di sintassi. È possibile utilizzare il vocabolario e la sintassi del BASIC per creare un insieme di istruzioni chiamate **programma**, che il computer può eseguire.

Per mezzo del BASIC è possibile comunicare con il Commodore 128 in due modi: all'interno di un programma o direttamente (al di fuori di un programma).

Modalità diretta

All'accensione il Commodore 128 è pronto ad accettare comandi BASIC in modalità **diretta**. In questa modalità i comandi vengono battuti da tastiera ed introdotti nel computer premendo il tasto RETURN. Il computer esegue tutti i comandi dati in modalità diretta subito dopo la pressione del tasto RETURN. La maggior parte dei comandi del Commodore 128 può essere utilizzata sia in modalità diretta sia all'interno di un programma.

Modalità programma

In **modalità programma** viene introdotta una serie di istruzioni per eseguire un compito specifico. Ogni istruzione è contenuta in una riga di **programma** sequenziale. Un'istruzione in un programma può essere composta da un massimo di 160 caratteri, che equivalgono a quattro righe intere di schermo con formato 40 colonne, e a due righe intere di schermo con formato 80 colonne.

Dopo aver battuto il programma, è possibile utilizzarlo immediatamente battendo il comando RUN seguito da RETURN. Si potrà inoltre memorizzare il programma su disco o nastro per mezzo del comando DSAVE (o SAVE). Lo si potrà quindi richiamare da disco o da nastro utilizzando il comando DLOAD (o LOAD). Questo comando copia il programma da disco o da nastro e lo trasferisce nella memoria del Commodore 128. Il programma potrà essere eseguito nuovamente battendo il comando RUN. Tutti questi comandi sono spiegati in dettaglio più avanti in questa sezione. Il

computer sarà quasi sempre utilizzato con i programmi, sia quelli creati personalmente, sia quelli già pronti per l'uso. Le uniche volte in cui si lavorerà in modalità diretta sarà quando si dovranno modificare i programmi utilizzando i comandi LIST, LOAD, SAVE e RUN. L'unica differenza tra la modalità diretta e la modalità programma è che nella prima i comandi non hanno il numero di riga.

Uso della tastiera

Di seguito viene mostrata la tastiera del Personal Computer Commodore 128.

Il BASIC è lo stesso per la modalità C64 e per la modalità C128. La maggior parte dei tasti e dei comandi può essere utilizzata per programmare in entrambe le modalità. I tasti evidenziati con l'ombreggiatura nella figura possono essere utilizzati in modalità C64. In modalità C128 si utilizzano tutti i tasti della tastiera.

Set di caratteri della tastiera

La tastiera del Commodore 128 offre due set differenti di caratteri:

- Lettere maiuscole e caratteri grafici
- Lettere maiuscole e minuscole

Nel formato a 80 colonne i due set sono disponibili simultaneamente. Si potranno così visualizzare sullo schermo 512 caratteri differenti. Nel formato a 40 colonne è possibile utilizzare un solo set alla volta.

All'accensione del Commodore 128 nel formato a 40 colonne, la tastiera è impostata sul set di caratteri maiuscolo/grafico, quindi tutte le lettere battute saranno maiuscole. Per passare da un set di caratteri all'altro, premere il tasto SHIFT contemporaneamente al tasto C= (tasto COMMODORE). Provare ad utilizzare i due set di caratteri accendendo il computer e premendo alcune lettere o alcuni caratteri grafici, quindi premendo i tasti SHIFT e C= (Commodore). Sullo schermo si potranno così notare i cambiamenti da maiuscolo a minuscolo delle lettere. Premere nuovamente SHIFT e C= per tornare al set di caratteri maiuscolo/grafico.

Uso dei tasti di comando

I tasti di COMANDO sono quei tasti che inviano i messaggi al computer. Alcuni tasti di comando (come il tasto RETURN) vengono usati da soli, mentre altri (come SHIFT, CTRL, C= e RESTORE) vengono utilizzati in combinazione con altri tasti. Di seguito viene spiegato l'uso di ciascun tasto di comando. I tasti utilizzati in modalità C128 vengono descritti nella Sezione 5.

Return

Premendo il tasto RETURN si otterrà l'invio di quanto battuto alla memoria del Commodore 128. Premendo questo tasto si otterrà anche lo spostamento del cursore (il piccolo rettangolo lampeggiante che indica dove apparirà il prossimo carattere battuto) alla riga sottostante.

Se viene battuto in modo errato un comando o se si introduce qualcosa che il computer non capisce, dopo aver premuto RETURN apparirà un messaggio (es. SYNTAX ERROR). Questo viene chiamato "Messaggio di Errore". Nell'Appendice A vengono elencati tutti i messaggi di errore insieme al modo per correggere gli errori stessi.

NOTA: Negli esempi dati in questo manuale, il simbolo illustrato di seguito indica che si deve premere il tasto RETURN:

RETURN

Shift

Sulla fila di tasti inferiore della tastiera sono presenti due tasti SHIFT, uno a destra ed uno a sinistra, come su una normale macchina per scrivere.

Il tasto SHIFT può essere usato in tre modi:

1. Con il set di caratteri maiuscolo/minuscolo, il tasto SHIFT viene utilizzato come il tasto delle maiuscole di una normale macchina per scrivere. Quando il tasto SHIFT è abbassato, si ottiene la lettera maiuscola o il carattere rappresentato nella metà superiore di un tasto a due caratteri.
2. Il tasto SHIFT può essere usato insieme ad altri tasti di comando per eseguire particolari funzioni.
3. Con la tastiera impostata sul tipo di carattere maiuscolo/grafico si utilizza il tasto SHIFT per stampare i simboli grafici o quei caratteri che appaiono sul lato frontale di alcuni tasti. Per ulteriori dettagli consultare il paragrafo "Visualizzazione dei caratteri grafici" alla fine di questa sezione.

Shift Lock

Quando lo si preme, questo tasto rimane abbassato. Quindi, tutti i caratteri battuti saranno o lettere maiuscole o i caratteri rappresentati nella metà superiore di un tasto a due caratteri. Per sbloccare il tasto SHIFT LOCK è sufficiente premerlo una seconda volta.

Spostamento del cursore

In modalità C128 il cursore può essere spostato sia utilizzando i quattro tasti freccia situati a destra al di sopra della tastiera principale, sia utilizzando i due tasti CRSR a destra sulla fila di tasti inferiore della tastiera principale.

Uso dei quattro tasti freccia

In modalità C128 il cursore può essere spostato in qualsiasi direzione utilizzando semplicemente il tasto freccia (situato sulla riga

di tasti superiore) che indica la posizione in cui si desidera muovere il cursore (questi tasti non possono essere utilizzati in modalità C64).

Uso dei tasti CRSR

Per spostare il cursore quando ci si trova nelle modalità C128 e C64 occorre utilizzare i due tasti situati a destra sulla fila di tasti inferiore della tastiera principale:

- Premendo il solo tasto **CRSR** (↑↓) il cursore si sposterà verso il **basso**.
- Premendo **CRSR** (↑↓) insieme al tasto SHIFT il cursore si sposterà verso **l'alto**.
- Premendo il solo tasto **CRSR** (←→) il cursore si sposterà verso **destra**.
- Premedo **CRSR** (←→) insieme al tasto SHIFT il cursore si sposterà verso **sinistra**.

Non è necessario battere più volte il tasto di movimento cursore per spostarsi di più spazi. È infatti sufficiente tenere il tasto premuto finchè il cursore non raggiunge la posizione desiderata.

Osservare come il cursore, dopo aver raggiunto il lato destro dello schermo, ritorna a capo, si sposta cioè all'inizio della riga sottostante. Quando si sposta il cursore verso sinistra, questo si muoverà sulla riga fino a raggiungere il margine sinistro, quindi salterà alla fine della riga precedente.

Si consiglia di acquisire dimestichezza con i tasti di movimento cursore in quanto questi tasti facilitano la programmazione. Con un po' di pratica si arriverà a spostare il cursore automaticamente senza nemmeno pensarci.

Inst/Del

Questo tasto ha due funzioni. INST serve per inserire, DEL per cancellare.

Inserimento di caratteri

Per inserire caratteri in una riga occorre usare il tasto SHIFT insieme al tasto INST/DEL. Se ad esempio non si sono battuti per errore dei caratteri in una riga, come nell'esempio seguente:

QUANDO ■ETE USCITI

per prima cosa, per poter introdurre i caratteri mancanti, si dovrà spostare il cursore sul punto in cui si è commesso l'errore:

QUANDO ■TE USCITI

quindi, tenendo premuto il tasto SHIFT, premere il tasto INST/DEL finchè non si otterrà spazio sufficiente per i caratteri da introdurre:

QUANDO ■ ETE USCITI

Notare che INST non sposta il cursore, ma aggiunge solamente dello spazio tra il cursore ed il carattere alla sua destra. Per completare la correzione, battere S e I:

QUANDO SIETE USCITI

Cancellazione di caratteri

Premendo il tasto DEL, il cursore si sposterà di uno spazio verso sinistra e cancellerà il carattere che si trova in quella posizione. Questo significa che quando si desidera cancellare qualcosa, occorre spostare il cursore a destra del carattere da cancellare. Se ad esempio si è commesso un errore di battitura come:

PRINT"ERROERE"

Si voleva battere la parola ERRORE, non ERROERE; per cancellare la E precedente l'ultima R, posizionare il cursore sulla R e premere il tasto DEL. Il carattere a destra del cursore (la R) si sposterà di uno spazio verso sinistra. Si otterrà così la parola corretta:

PRINT"ERRORE"

Uso contemporaneo di INST e DEL

Nel corso di un'operazione di correzione è possibile usare contemporaneamente INST e DEL. Per prima cosa spostare il cursore sui caratteri da correggere e premere il tasto INST/DEL per cancellare i caratteri stessi. Quindi, premere contemporaneamente SHIFT e INST/DEL per aggiungere lo spazio necessario. A questo punto battere le correzioni. È possibile battere direttamente sopra i caratteri da cancellare e quindi usare INST per aggiungere lo spazio necessario.

Control

Il tasto Control viene utilizzato in combinazione con altri tasti per ottenere funzioni speciali chiamate funzioni di controllo. Per eseguire una di queste funzioni, tenere premuto il tasto Control con-

temporaneamente ad un altro tasto. Una lista completa delle sequenze di contollo viene fornita nell'Appendice I. Le funzioni di controllo vengono spesso utilizzate nei programmi già pronti per l'uso come ad esempio nei programmi di elaborazione testi.

La funzione di controllo utilizzata maggiormente è quella di impostazione colore per i caratteri ed il cursore. Per selezionare un colore, tenere premuto il tasto CTRL contemporaneamente ad uno dei tasti numerici (da 1 a 8) situati nella fila superiore della tastiera. Vi sono altri otto colori a disposizione, che possono essere selezionati per mezzo del tasto C= , come verrà spiegato più avanti.

Run/Stop

Questo tasto ha due funzioni. In alcuni casi la funzione RUN di questo tasto viene ottenuta premendo SHIFT contemporaneamente a RUN/STOP. È inoltre possibile utilizzare la funzione STOP di questo tasto per interrompere un programma o una stampa in fase di esecuzione del programma stesso. Comunque, nella maggior parte dei programmi già pronti la funzione STOP del tasto RUN/STOP viene intenzionalmente disattivata. Questo per evitare che l'utente interrompa un programma prima che questo venga eseguito completamente. Se questa funzione non fosse disattivata, si correrebbe il rischio di perdere dati arrestando il programma.

Restore

Il tasto RESTORE viene utilizzato insieme al tasto RUN/STOP per riportare il computer allo stato iniziale.

In molti programmi già pronti questa funzione viene disattivata per lo stesso motivo per cui viene disattivata la funzione del tasto RUN/STOP: per evitare la perdita di dati preziosi.

CLR/Home

CLR significa cancellare. HOME è il nome dato all'angolo in alto a sinistra sullo schermo. Premendo questo tasto da solo il cursore ritorna alla posizione HOME, premendolo invece contemporaneamente al tasto SHIFT si ottiene la cancellazione dell'intero schermo e il cursore si sposta alla posizione HOME.

Tasto Commodore (C=)

Il tasto C= (chiamato anche tasto COMMODORE) ha molte funzioni, tra cui:

1. Il tasto C= permette di passare dal set di caratteri maiuscolo/minuscolo e il set di caratteri maiuscolo/grafico. Per passare da un set all'altro premere contemporaneamente i tasti C= e SHIFT.
2. Da una delle due modalità il tasto C= agisce come tasto shift per ottenere i simboli grafici rappresentati sul lato sinistro di ciascun tasto. Premere semplicemente il tasto C= insieme al tasto grafico desiderato.
3. Il tasto C= permette inoltre di utilizzare una serie di otto colori addizionali per il cursore. Per ottenere questi colori, tenere premuto il tasto C= insieme ad uno dei tasti numerici (da 1 a 8) situati sulla fila superiore della tastiera.
4. Per rallentare la visualizzazione in scorrimento di un programma, tenere premuto il tasto C=. In questo modo lo scorrimento viene notevolmente rallentato. Per tornare allo scorrimento a velocità normale, rilasciare il tasto.
5. Tenendo premuto il tasto C= mentre si accende il computer, si avrà accesso alla modalità C64.

Tasti funzione

I quattro tasti situati al di sopra del tastierino numerico (contrassegnati F1, F3, F5 e F7 sul lato superiore e F2, F4, F6 e F8 sulla parte frontale) sono chiamati **tasti funzione**. In entrambe le modalità C128 e C64 i tasti funzione possono essere programmati (vedere le descrizioni del comando KEY nella Sezione 5 del Capitolo II e nel Capitolo V, ENCICLOPEDIA BASIC 7.0). Questi tasti vengono spesso utilizzati nei programmi già pronti per poter eseguire un'operazione con la pressione di un unico tasto.

Visualizzazione dei caratteri grafici

Per visualizzare il simbolo grafico raffigurato sulla parte frontale di un tasto, tenere premuto il tasto SHIFT contemporaneamente al tasto interessato. I caratteri grafici a destra possono essere visualizzati solo se la tastiera è impostata sul set di caratteri maiuscolo/grafico (il set di caratteri attivo all'accensione).

Per poter visualizzare il carattere grafico sulla sinistra del lato frontale di un tasto, tenere premuto il tasto C= insieme al tasto interessato. Questo carattere potrà essere visualizzato in entrambi i set di caratteri della tastiera.

Regole di battitura dei programmi in linguaggio BASIC

I programmi in linguaggio BASIC potranno essere battuti ed utilizzati anche senza conoscere il BASIC. Si dovrà prestare attenzione alla battitura in quanto un errore potrebbe causare la non accettazione da parte del computer dell'informazione introdotta. Di seguito vengono dati dei consigli per ridurre errori di battitura o di copiatura di un listato del programma.

1. Lo spazio tra le parole non è importante; ad esempio, FORT=1TO10 è uguale a FOR T=1 TO 10. Una parola chiave BASIC però non deve essere interrotta da spazi (consultare nell'Enciclopedia BASIC 7.0, Capitolo V, la lista delle parole chiavi BASIC).
2. Tutti i caratteri possono essere racchiusi tra virgolette. Alcuni caratteri però avranno funzioni speciali se racchiusi tra virgolette. Queste funzioni saranno spiegate più avanti in questo manuale.
3. Prestare attenzione alla punteggiatura. Le virgole, i due punti e i punti e virgola hanno funzioni speciali che verranno illustrate più avanti in questa parte.
4. Premere sempre il tasto RETURN (indicato con RETURN in questo manuale) dopo aver terminato una riga numerata.
5. Non superare mai i 160 caratteri in una riga di programma. 160 caratteri corrispondono a 4 righe complete di schermo se in formato a 40 colonne oppure a due righe complete di schermo se in formato a 80 colonne. Vedere la Sezione 8 per ulteriori dettagli sui formati a 40 o 80 colonne.
6. Fare attenzione a non confondere la lettera l con il numero 1 e la lettera O con il numero 0.
7. Il computer ignora tutto quello che si trova dopo le lettere REM in una riga di programma. REM sta per note. L'istruzione REM serve ad inserire commenti nel programma per dare informazioni su cosa accadrà ad un punto specifico del programma.

Durante la battitura degli esempi e dei programmi illustrati in questa sezione del manuale, seguire i consigli riportati sopra.

Avviamento - Il comando PRINT

Il comando PRINT comunica al computer di visualizzare le informazioni sullo schermo. Si possono stampare sia numeri sia testo (lettere), ma per ognuno dei due casi esistono delle regole che verranno illustrate di seguito.

Stampa di numeri

Per stampare numeri utilizzare il comando PRINT seguito dal numero o numeri che si desiderano stampare. Battere:

PRINT 5

quindi premere il tasto RETURN. Notare che il numero 5 è visualizzato sullo schermo.

Ora battere, seguito da RETURN:

PRINT 5,6

In questo comando PRINT la virgola informa il Commodore 128 che si desidera stampare più di un numero. Quando il computer trova delle virgole in stringhe di numeri di un'istruzione PRINT, ogni numero dopo la virgola sarà stampato 10 spazi a destra del numero che lo precede. Se non si desiderano gli spazi addizionali, utilizzare un punto e virgola (;) invece di una virgola nell'istruzione PRINT. Il punto e virgola informa il computer di stampare i numeri uno accanto all'altro. Quando il numero sarà stampato, davanti ad esso vi sarà uno spazio o un segno di meno e dopo di esso un carattere di salto. Battere i seguenti esempi ed osservarne i risultati:

PRINT 5;6 RETURN

PRINT 100;-200;300;-400;500 RETURN

Uso del punto interrogativo per abbreviare il comando PRINT

È possibile utilizzare un punto interrogativo (?) per abbreviare il comando PRINT. Molti esempi in questa sezione del manuale usano il simbolo ? al posto della parola PRINT. La maggior parte dei comandi BASIC può essere abbreviata. Le abbreviazioni dei comandi BASIC sono illustrate nell'appendice K di questo manuale.

Stampa del testo

Dopo aver appreso come stampare i numeri, si vedrà come stampare il testo. Tutte le parole o i caratteri che si vogliono visualizzare devono essere battuti sullo schermo con le virgolette all'inizio e alla fine della stringa di caratteri. **Stringa è il nome BASIC per una qualsiasi serie di caratteri racchiusi tra virgolette**. Le virgolette si ottengono premendo SHIFT insieme al tasto numerico 2 situato sulla fila superiore della tastiera (non il tasto 2 del tastierino numerico). Battere:

?"COMMODORE 128" RETURN

?"4*5" RETURN

Notare che premendo RETURN il computer visualizza i caratteri racchiusi tra virgolette. Notare inoltre che nel secondo esempio non è stato stampato il risultato di 4*5 in quanto questi numeri sono stati trattati come stringa e non come un'operazione matematica. Per calcolare il risultato di 4*5, utilizzare il comando seguente:

? 4*5 RETURN

È possibile stampare qualsiasi stringa di caratteri utilizzando il comando PRINT e racchiudendo i caratteri da stampare tra virgolette. Per combinare testo e operazioni matematiche in un singolo comando PRINT utilizzare un formato come il seguente:

?"4*5="4*5 RETURN

Notare come il computer stampa i caratteri tra virgolette, esegue l'operazione e stampa il risultato. La posizione del testo e dell'operazione non ha alcuna importanza, infatti è possibile utilizzare entrambi più volte in un comando PRINT. Battere l'istruzione seguente:

?4*(2+3)"è uguale a"4*5 RETURN

Notare che sullo schermo vengono visualizzati anche gli spazi all'interno delle virgolette. Battere:

?"   QUI" RETURN

Stampa in diversi colori

Il Commodore 128 può visualizzare 16 diversi colori, che possono essere facilmente cambiati. Per ottenere un colore è sufficiente premere il tasto CTRL insieme ad un tasto numerato da uno a otto sulla fila superiore della tastiera principale. Il cursore sarà di un colore diverso a seconda del tasto numerico premuto. Tutti i caratteri dopo il cursore saranno del colore selezionato. Tenendo premuto il tasto Commodore insieme ad un tasto numerato compreso uno e otto si otterranno otto colori addizionali.

La tabella 3-1 elenca i colori disponibili in modalità C128, per i formati a 40 e 80 colonne. La tabella indica inoltre la sequenza di tasti (tasto CONTROL più un tasto numerico oppure il tasto C= più un tasto numerico) utilizzata per specificare un dato colore.

Control +	Colore	C=	Colore
1	Nero	1	Arancione
2	Bianco	2	Marrone
3	Rosso	3	Rosso chiaro
4	Azzurro	4	Grigio scuro
5	Porpora	5	Grigio
6	Verde	6	Verde chiaro
7	Blu	7	Blu chiaro
8	Giallo	8	Grigio chiaro

Formato a 40 colonne

Control +	Colore	C=	Colore
1	Nero	1	Porpora scuro
2	Bianco	2	Marrone
3	Rosso scuro	3	Rosso chiaro
4	Azzurro chiaro	4	Azzurro scuro
5	Porpora chiaro	5	Grigio
6	Verde scuro	6	Verde chiaro
7	Blu scuro	7	Blu chiaro
8	Giallo chiaro	8	Grigio chiaro

Formato a 80 colonne

Uso dei tasti cursore all'interno di virgolette con il comando PRINT

Utilizzando i tasti di movimento cursore all'interno delle virgolette, sullo schermo vengono visualizzati i caratteri grafici per rappresentare i tasti. Questi caratteri non saranno più presenti sullo schermo quando verrà premuto il tasto RETURN. Battere un punto interrogativo (?), le virgolette (tasto numerico 2 tenendo premuto SHIFT), quindi premere uno dei tasti cursore "giù" 10 volte, battere la parola "QUI" e chiudere le virgolette. Dovrebbe apparire quanto segue:

?" QQQQQQQQQQ QUI"

A questo punto premere RETURN. Il Commodore 128 visualizza 10 righe vuote e, sull'undicesima riga, visualizzerà "QUI". Come mostra l'esempio precedente, è possibile comunicare al computer, per mezzo dei tasti di controllo cursore all'interno di virgolette, il punto in cui dovrà stampare sullo schermo.

Come iniziare a programmare

Fino a questo momento, la maggior parte dei comandi trattati venivano eseguiti in modalità DIRETTA. Vale a dire che il comando veniva eseguito subito dopo la pressione del tasto RETURN. Comunque, la maggior parte dei comandi e delle funzioni BASIC possono essere utilizzate all'interno di programmi.

Che cos'è un programma

Un programma è una serie di istruzioni BASIC numerate che indicano al computer le operazioni da svolgere. Queste istruzioni numerate vengono chiamate **statement** o **righe di comando**.

Numeri di riga

Le righe di un programma sono numerate in modo che il computer possa riconoscere l'ordine in cui devono essere eseguite. Il computer esegue le righe di programma seguendo l'ordine numerico, a meno che il programma non indichi diversamente. Per la numerazione delle righe possono essere usati i numeri da 0 a 63999. In un numero di riga non usare **mai** la virgola.

Molti dei comandi utilizzati fino a questo momento sono stati utilizzati in modalità DIRETTA, ma possono essere contenuti anche in istruzioni di programma. Ad esempio, battere:

10 ?"COMMODORE 128" RETURN

Notare come il computer non abbia visualizzato sullo schermo COMMODORE 128 dopo la pressione del tasto RETURN, come invece avrebbe fatto se si fosse utilizzato il comando PRINT in modalità DIRETTA. Questo perchè il numero 10 prima del simbolo PRINT (?) comunica al computer che si sta introducendo un programma in BASIC. Il computer memorizza l'istruzione numerata ed attende le introduzioni successive.

Ora battere RUN e premere RETURN. Il computer visualizzerà le parole COMMODORE 128. Questa operazione non ha lo stesso effetto dell'uso del comando PRINT in modalità DIRETTA. Quello che è successo ora è che AVETE SCRITTO ED ESEGUITO IL VOSTRO PRIMO PROGRAMMA BASIC per quanto corto possa essere. Il programma è ancora presente nella memoria del computer e lo si potrà eseguire ogni volta che lo si desidera.

Visualizzazione del programma - Il comando LIST

Il programma composto da una riga visto precedentemente si trova ancora nella memoria del C128. A questo punto, cancellare lo schermo premendo contemporaneamente i tasti SHIFT e CLR/HOME. Lo schermo sarà così vuoto. Il linguaggio BASIC prevede un comando per listare il programma quando si ha necessità di controllare che questo si trovi ancora nella memoria del computer - il comando LIST.

Battere LIST e premere RETURN. Il C128 visualizza:

10 PRINT "COMMODORE 128"

READY.

Ogni volta che si desidera visualizzare tutte le righe di un programma, battere LIST. Questa funzione è utile soprattutto quando si eseguono delle modifiche e ci si vuole accertare che le nuove righe siano state registrate nella memoria del computer. In risposta al comando il computer visualizza la versione modificata della riga, righe o programma. Di seguito vengono date delle regole per l'uso del comando LIST:

- Per visualizzare solo la riga N, battere LIST N e premere RETURN. Sostituire N con il numero di riga desiderato.
- Per visualizzare dalla riga N alla fine del programma, battere LIST N - e premere RETURN.
- Per visualizzare le righe dall'inizio del programma fino alla riga N, battere LIST-N e premere RETURN.

- Per visualizzare dalla riga N1 alla riga N2 compresa, battere LIST N1-N2 e premere RETURN.

Un semplice loop - L'istruzione GOTO

I numeri di riga in un programma hanno un altro scopo oltre a quello di sistemare i comandi nell'ordine corretto per il computer. Servono come riferimento al computer in caso si desideri eseguire la stessa riga di comando più volte in un programma. Utilizzare il comando GOTO per comunicare al computer di portarsi su una riga e di eseguire il comando (o comandi) in essa contenuto. A questo punto battere:

20 GOTO 10

La pressione di RETURN alla fine della riga 20 causa la sua aggiunta al programma nella memoria del computer.

Notare che la prima riga è stata numerata 10 e la seconda 20. È infatti utile numerare le righe di programma ad incrementi di 10 (cioè 10, 20, 30, 40 ecc.) per poter aggiungere in seguito ulteriori righe di programma tra una riga e l'altra. Queste righe aggiuntive possono essere numerate con multipli di cinque (15, 25...), di uno (1, 2...) - o con qualsiasi numero intero - per mantenere le righe nell'ordine corretto (vedere i comandi RENUMBER e AUTO nell'Encicolopedia BASIC).

Battere RUN e premere RETURN. Le parole COMMODORE 128 verranno stampate su tutto lo schermo. Per fermare la visualizzazione del messaggio sullo schermo, premere il tasto RUN/STOP situato a sinistra sulla tastiera.

Le due righe che sono state battute formano un semplice programma che si ripeterà all'infinito in quanto la seconda riga ordina al computer di tornare alla prima riga. Il programma continuerà finchè non verrà fermato o non si spegnerà il computer.

Ora battere **LIST** **RETURN**. Lo schermo visualizzerà:

10 PRINT "COMMODORE 128"
20 GOTO 10
READY.

Il programma si trova ancora in memoria e può essere eseguito nuovamente. Questa è una differenza importante tra la modalità PROGRAMMA e la modalità DIRETTA. Dopo che il comando viene eseguito in modalità DIRETTA non è più presente nella me-

moria del computer. Notare che anche se si è utilizzato il simbolo ? al posto dell'istruzione PRINT, il computer lo ha convertito nel comando completo. Questo succede sempre quando si lista un comando che era stato abbreviato in fase di introduzione.

Cancellazione della memoria del computer - Il comando NEW

Quando si desidera riprendere dall'inizio o cancellare un programma BASIC dalla memoria del computer, battere NEW e premere RETURN. Questo comando cancella la memoria BASIC del computer, cioè l'area in cui sono memorizzati i programmi.

Utilizzo del colore in un programma

Per selezionare il colore all'interno di un programma si dovrà includere l'informazione di selezione colore in un'istruzione PRINT. Ad esempio, cancellare la memoria del computer battendo NEW e premendo RETURN, quindi battere la frase seguente, lasciando uno spazio tra ogni lettera della parola tra virgolette:

10 PRINT "S P E C T R U M" RETURN

Ora battere nuovamente la riga 10 tenendo però premuto il tasto CTRL e battendo il tasto 1 direttamente dopo l'introduzione delle prime virgolette. Rilasciare il tasto CTRL e battere la "S". Ora tenere premuto nuovamente il tasto CTRL e premere il tasto 2. Rilasciare il tasto CTRL e battere la "P". Quindi tenere premuto il tasto CTRL e premere il tasto 3 e così via fino ad aver battuto tutte le lettere della parola SPECTRUM ed aver selezionato un colore tra ogni lettera. Premere i tasti SHIFT e 2 per battere le virgolette di chiusura e premere il tasto RETURN. Ora battere RUN e premere il tasto RETURN. Il computer visualizza la parola SPECTRUM con ogni lettera di un colore diverso. Battere LIST e premere il tasto RETURN. Notare i caratteri grafici che appaiono nell'istruzione PRINT sulla riga 10. Questi caratteri indicano al computer il colore desiderato per ogni lettera e non appariranno quando il Commodore 128 stamperà la parola SPECTRUM in diversi colori.

I caratteri di selezione colore, chiamati caratteri di controllo, nell'istruzione PRINT della riga 10 comunicano al Commodore 128 di cambiare i colori. Il computer stampa i caratteri successivi con il nuovo colore finchè non incontrerà il prossimo carattere di selezione colore. Mentre i caratteri tra virgolette vengono stampati come appaiono sullo schermo, i caratteri di controllo vengono visualizzati solo in seguito ad un comando LIST.

Modifica del programma

I paragrafi seguenti aiuteranno nell'introduzione dei programmi e nelle correzioni ed aggiunte ai programmi stessi.

Cancellazione di una riga da un programma

Utilizzare il comando LIST per visualizzare il programma precedentemente battuto. Ora battere 10 e premere RETURN. Con questa operazione si è cancellata la riga 10 dal programma. Listare il programma e controllare. Se la vecchia riga 10 è ancora visualizzata sullo schermo, spostare il cursore su un punto qualsiasi della riga 10. Premendo RETURN la riga 10 sarà riportata nella memoria del computer.

Duplicazione di una riga

Tenere premuto il tasto SHIFT contemporaneamente al tasto CLR/HOME in alto a destra sulla tastiera. Questa operazione cancellerà l'intero schermo. Ora, listare il programma. Muovere il cursore verso l'alto fino a portarlo sullo "0" della riga 10. Battere un 5 e premere RETURN. In questo modo si è duplicata (cioè copiata) la riga 10. Alla riga duplicata viene assegnato il numero 15. Battere LIST e premere RETURN per visualizzare il programma con le righe duplicate.

Sostituzione di una riga

È possibile sostituire un'intera riga battendo il vecchio numero di riga seguito dal testo della riga nuova e premendo infine RETURN. La vecchia versione della riga verrà cancellata dalla memoria e sostituita dalla nuova riga subito dopo la pressione di RETURN.

Modifica di una riga

Se si desidera aggiungere del testo all'interno di una riga, spostare il cursore sul carattere o sullo spazio immediatamente seguente il punto in cui si desidera effettuare l'inserimento, quindi premere contemporaneamente i tasti SHIFT e INST/DEL fino ad avere spazio sufficiente per inserire i nuovi caratteri.

Provare l'esempio seguente. Cancellare la memoria del computer battendo NEW seguito da RETURN, quindi battere:

10 ? "IL MIO 128 È ECCEZIONALE" RETURN

Si aggiungerà ora la parola COMMODORE prima di 128. Spostare il cursore sul numero "1" di 128. Premere SHIFT contemporaneamente al tasto INST/DEL fino ad avere spazio sufficiente per la parola COMMODORE (compreso uno spazio dopo la E). Quindi battere la parola COMMODORE.

Se si desidera cancellare alcuni caratteri all'interno di una riga (compresi gli spazi vuoti), spostare il cursore sul carattere che segue la parte da cancellare, quindi tenere premuto il tasto INST/DEL. Il cursore si muoverà verso sinistra ed i caratteri o gli spazi saranno cancellati finchè si terrà premuto il tasto INST/DEL.

Operazioni matematiche

Il comando PRINT può essere utilizzato per eseguire calcoli come addizione, sottrazione, moltiplicazione, divisione ed elevazione a potenza. Il calcolo viene battuto dopo il comando PRINT.

Addizione e sottrazione

Provare i seguenti esempi:

PRINT 6 + 4 RETURN
PRINT 50 – 20 RETURN
PRINT 10 + 15 – 5 RETURN
PRINT 75 – 100 RETURN
PRINT 30 + 40,55 – 25 RETURN
PRINT 30 + 40;55 – 25 RETURN

Notare che il quarto calcolo (75—100) dà un risultato negativo. Notare inoltre che è possibile dare al computer più calcoli con un solo comando PRINT. Nel comando si può utilizzare una virgola o un punto e virgola a seconda che si desiderino i risultati tabulati o uno accanto all'altro.

Moltiplicazione e divisione

L'asterisco (*) a destra sulla tastiera è il simbolo utilizzato dal Commodore 128 per la moltiplicazione. Il tasto barra (/) invece, situato accanto al tasto SHIFT viene utilizzato per la divisione.

Provare gli esempi seguenti:

PRINT 5*3 RETURN
PRINT 100/2 RETURN

Elevazione a potenza

Per elevare un numero a una potenza, utilizzare il tasto freccia verso l'alto (↑), situato vicino all'asterisco sulla tastiera. Se si desidera elevare un numero ad una potenza usare il comando PRINT seguito dal numero, la freccia verso l'alto e la potenza, in questo stesso ordine. Ad esempio, per trovare il risultato di 3 alla seconda, battere:

PRINT 3↑2 RETURN

Priorità delle operazioni

Si è visto in precedenza come combinare addizione e sottrazione nello stesso comando PRINT. Combinando però operazioni di moltiplicazione o divisione con operazioni di addizione o sottrazione il risultato potrebbe non essere quello desiderato. Ad esempio, battere:

PRINT 4 + 6/2 RETURN

Se lo scopo era di dividere 10 per 2, ci si sorprenderà nel vedere che il computer dà come risultato 7. La ragione di questo risultato è che il computer esegue le operazioni di moltiplicazione e divisione prima delle operazioni di addizione e sottrazione. La moltiplicazione e la divisione hanno la priorità sulla addizione e la sottrazione. L'ordine di introduzione delle operazioni non ha alcuna importanza. Con il computer l'ordine di esecuzione delle operazioni matematiche viene detto priorità delle operazioni.

L'esponenziazione, o elevazione di un numero a potenza, ha la precedenza su tutte le quattro operazioni matematiche. Ad esempio battendo:

PRINT 16/4↑2 RETURN

il Commodore 128 risponde 1 in quanto eleva al quadrato il 4 prima di dividerlo per 16.

Uso delle parentesi per la definizione della priorità delle operazioni

È possibile comunicare al Commodore 128 quale operazione dovrà essere eseguita per prima racchiudendo l'operazione stessa tra parentesi nel comando PRINT. Prendendo in considerazione il primo esempio precedente, se si desidera eseguire per prima l'o-

perazione di addizione e quindi quella di divisione, si dovrà battere:

PRINT (4 + 6)/2 RETURN

Questo darà come risultato 5.

Se si desidera che il computer esegua per prima cosa l'operazione di divisione e quindi l'elevazione al quadrato nel secondo esempio precedente, battere:

PRINT (16/4)↑2 RETURN

Il risultato sarà 16.

Se non si utilizzano le parentesi, il computer eseguirà le operazioni seguendo le regole spiegate in precedenza. Quando tutte le operazioni hanno la stessa priorità vengono eseguite da sinistra a destra. Ad esempio, battendo:

PRINT 4*5/10*6 RETURN

il risultato sarà 12 (4*5 = 20...20/10 = 2...2*6 = 12) perchè le operazioni vengono eseguite da sinistra a destra. Se si desidera dividere 4*5 per 10*6, battere:

PRINT (4*5)/(10*6) RETURN

Il risultato ora sarà .333333333.

Costanti, variabili e stringhe

Costanti

Le costanti sono valori numerici permanenti: cioè il loro valore non viene modificato nel corso di un'equazione o di un programma. Ad esempio, il numero 3 è una costante, come un qualsiasi altro numero. L'istruzione seguente mostra come il computer utilizza le costanti:

10 PRINT 3

La risposta sarà sempre 3 indipendentemente da quante volte la riga di programma viene eseguita.

Variabili

Le varibili sono valori che possono variare nel corso di un'equazione o di un'istruzione di programma. Nella memoria BASIC del computer una parte è dedicata ai caratteri (numeri, lettere e simboli) che vengono utilizzati in un programma. Questo tipo di me-

moria è paragonabile a scomparti di memorizzazione nel computer che memorizzano le informazioni di un programma; questa parte nella memoria del computer viene chiamata memoria variabile. Battere il seguente programma:

```
10 X = 5
20 ?X
```

Ora eseguire il programma e notare come il computer visualizza un 5 sullo schermo. Nella riga 10 si era comunicato al computer che la lettera X rappresenta il numero 5 nel corso dell'intero programma. La lettera X è detta variabile in quanto il valore di X varia a seconda del valore che si troverà a destra del simbolo di uguale. Questa è un'istruzione di assegnazione, infatti ora esiste una sezione di memorizzazione definita X nella memoria del computer ed il numero 5 è stato assegnato a questa sezione. Il segno = indica al computer che tutto quello che si troverà a destra di questo simbolo sarà assegnato ad una sezione di memorizzazione (locazione di memoria) definita con la lettera X a destra del segno di uguale.

Il nome variabile alla destra del segno = può essere composto da una o due lettere oppure da una lettera e un numero (il primo carattere DEVE essere una lettera). Il nome può essere più lungo, ma il computer prenderà in considerazione solo i primi due caratteri. Ciò significa che i nomi PA e PARTE faranno riferimento alla stessa sezione di memorizzazione. Inoltre, le parole utilizzate per i comandi (LOAD, RUN, LIST ecc.) o per le funzioni (INT, ABS, SQR, ecc.) BASIC non possono essere utilizzate come nomi in un programma. Consultare l'Enciclopedia BASIC nel Capitolo 5 per verificare se un nome di variabile è una parola chiave BASIC. Notare che il simbolo = nelle istruzioni di assegnazione non equivale al simbolo matematico "uguale", ma significa allocare una variabile (sezione di memorizzazione) ed assegnare un valore alla variabile stessa.

Nel programma di esempio di cui sopra, il valore della variabile X sarà sempre 5. È possibile assegnare un risultato di una operazione ad una variabile battendo i calcoli a destra del segno di =. Per identificare le costanti si potrà utilizzare del testo insieme alle costanti in un'istruzione PRINT. Battere NEW e premere RETURN per cancellare la memoria del Commodore 128, quindi battere il

seguente programma:

10 A = 3*100
20 B = 3*200
30 ?"A È UGUALE A "A
40 ?"B È UGUALE A "B

A questo punto, nella memoria del computer sono presenti due variabili, definite A e B che contengono rispettivamente i numeri 300 e 600. Se nel corso di un programma si desidera cambiare il valore di una variabile, sarà sufficiente introdurre nel programma un'altra istruzione di assegnazione. Aggiungere le seguenti righe al programma precedente ed eseguirlo:

50 A = 900*30/10
60 B = 95 + 32 + 128
70 GOTO 30

Per arrestare il programma, premere il tasto STOP.

Ora listare il programma e seguire le fasi svolte dal computer. Per prima cosa il computer ha assegnato alla lettera A il valore a destra del segno di = nella riga 10. Quindi ha fatto lo stesso per la lettera B nella riga 20, ha stampato i messaggi contenuti nelle righe 30 e 40 che danno i valori di A e B. Infine ha assegnato un nuovo valore ad A e B nelle righe 50 e 60. I vecchi valori vengono sostituiti e non possono essere recuperati a meno che il computer non esegua nuovamente le righe 10 e 20. Quando il computer viene inviato alla riga 30 per ristampare i valori di A e B stamperà i nuovi valori calcolati nelle righe 50 e 60. Queste ultime due righe assegnano nuovamente gli stessi valori a A e B e la riga 70 rimanda il computer alla riga 30. Questo viene chiamato un loop infinito, in quanto le righe da 30 a 70 saranno eseguite continuamente finchè non sarà premuto il tasto RUN/STOP per arrestare il programma. Nei prossimi due capitoli verranno spiegati altri tipi di loop.

Stringhe

Una stringa è un carattere o un gruppo di caratteri racchiusi tra virgolette. Questi caratteri vengono memorizzati nella memoria del computer come variabili come avviene per le variabili numeriche. Per rappresentare le stringhe si possono utilizzare nomi variabili, esattamente come vengono utilizzati per rappresentare i numeri. Battendo il simbolo di dollaro ($) dopo il nome variabile, si informerà il computer che il nome è per una variabile stringa e non per una variabile numerica.

Battere NEW e premere RETURN per cancellare la memoria del computer, quindi battere il programma seguente:

```
10 A$ ="COMPUTER"
20 X = 128
30 B$ = "COMMODORE"
40 Y = 1
50 ? "IL "A$;X;B$" È IL NUMERO "Y
```

Come si è visto, è possibile stampare variabili numeriche e di stringa nella stessa istruzione. Esercitarsi ora con le variabili creando piccoli programmi.

Il valore di una variabile può essere stampato in modalità DIRETTA, dopo l'esecuzione del programma. Battere ?A$;B$;X;Y dopo l'esecuzione del programma precedente e notare che i tre valori variabili sono ancora presenti nella memoria del computer.

Per cancellare questa area dalla memoria BASIC, senza però cancellare il programma, utilizzare il comando CLR. Battere CLR <RETURN>. Tutte le costanti, le variabili e le stringhe saranno cancellate. Quando però si batterà LIST, si potrà notare che il programma è ancora presente in memoria. Il comando NEW visto precedentemente cancella sia il programma sia le variabili.

Esempio di programma

Di seguito viene dato un esempio di programma che contiene molte delle tecniche e dei comandi illustrati in questa sezione.

Questo programma calcola la media di tre numeri (X, Y e Z) e stampa sullo schermo i loro valori e le loro medie. Per modificare i valori delle variabili è possibile modificare il programma variando i calcoli nelle righe da 10 a 30. La riga 40 somma le variabili e divide per 3 per ottenere la media. Notare l'uso delle parentesi per comunicare al computer di sommare i numeri prima di dividerli.

SUGGERIMENTO: Quando si utilizza più di una coppia di parentesi in un'istruzione, si consiglia di contare le parentesi aperte e chiuse per accertarsi che siano di numero pari.

```
10 X = 46
20 Y = 72
30 Z = 114
40 A = (X + Y + Z)/3
50 ?"LA MEDIA DI "X;Y;" E "Z; "È"A;
60 END
```

Memorizzazione e riutilizzo di un programma

Dopo la creazione di un programma, lo si potrà memorizzare definitivamente per poterlo richiamare e riutilizzare in seguito. Per questa operazione occorre un disk drive Commodore o il Datassette Commodore 1530.

Verranno spiegati diversi comandi per poter comunicare con il computer e il disk drive o Datassette. Questi comandi sono composti da una parola di comando seguita da diversi parametri. I parametri sono lettere, parole o simboli in un comando che forniscono informazioni specifiche al computer, come ad esempio il nome di file o una variabile numerica che specifichi un numero di dispositivo. Ogni comando può avere più parametri. Ad esempio, i parametri del comando di formattazione del disco comprendono un nome per il disco e un numero identificativo o codice, più diversi altri parametri. I parametri vengono utilizzati in quasi tutti i comandi BASIC; alcuni sono variabili, ed altri sono costanti. Questi sono i parametri che forniscono le informazioni di disco al C128 e al disk drive:

Parametri di gestione disco

nome del disco	- nome identificativo di 16 caratteri qualsiasi fornito dall'utente.
nome di file	- nome identificativo di 16 caratteri qualsiasi fornito dall'utente.
codice id.	- codice identificativo di 2 caratteri qualsiasi fornito dall'utente.
numero del drive	- utilizzare 0 per un drive singolo, 0 o 1 per un drive doppio.
numero di dispositivo	- numero preassegnato ad un dispositivo periferico. Ad esempio, il numero di dispositivo di un disk drive Commodore è 8.

Formattazione di un disco - Il comando HEADER

Per memorizzare i programmi su un disco nuovo (o vergine), per prima cosa occorre preparare il disco in modo che possa ricevere i dati. Questa operazione viene chiamata "formattazione" del disco. **NOTA**: Assicurarsi di avere acceso il disk drive prima di inserire il disco.

Il processo di formattazione divide il disco in sezioni chiamate tracce e settori. Viene creato un indice (directory). Ogni volta che

un programma viene memorizzato sul disco, il nome assegnato al programma verrà aggiunto alla lista.

Il Commodore 128 ha due tipi di comandi di formattazione. Uno può essere utilizzato solo in modalità C128, ed uno può essere utilizzato sia in modalità C64 sia in C128. I paragrafi seguenti descrivono i comandi di formattazione per la modalità C128. Vedere il Capitolo III, che descrive la modalità C64, per maggiori dettagli sulla programmazione e la gestione dei dischi in C64.

Il comando che formatta un dischetto è il comando HEADER e può avere un formato esteso o breve. Per formattare un disco vergine (nuovo), si DEVE utilizzare la forma estesa, come indicato di seguito:

HEADER "nome disco",I id. [,Dnumero drive] [,[ON]U numero dispositivo]

Dopo la parola HEADER, battere un nome a scelta per il disco, racchiuso tra virgolette. Il nome può essere composto da un massimo di 16 caratteri. I nomi di disco dovranno essere identificativi per i lavori memorizzati sul disco.

Dopo il nome disco battere una virgola e la lettera "I," quindi un identificativo di due caratteri, seguito da una virgola. L'identificativo del disco non deve essere composto necessariamente da numeri; possono essere utilizzate anche lettere. Per i dischi si può utilizzare un sistema di codificazione a numeri consecutivi, ad esempio A1, A2, B1, B2.

Se si possiede un disk drive singolo, premere RETURN a questo punto. Il Commodore 128 assumerà automaticamente il numero di drive 0 ed il numero di dispositivo 8. Se si possiede più di un disk drive o un drive doppio, specificare questi parametri.

Il parametro successivo nel comando seleziona il numero di drive. Premere il tasto "D" e, se si possiede un disk drive singolo, premere il tasto zero seguito da una virgola. Nelle unità doppie i drive sono identificati con 0 e 1. Il parametro del numero di dispositivo inizia con la lettera U, quindi premere il tasto "U" seguito dal numero di dispositivo preassegnato per il disk drive Commodore, 8.

Ecco un esempio del formato esteso del comando HEADER:

HEADER "RECS",IA1,D0,U8 RETURN

Questo comando esegue la formattazione chiamando il dischetto RECS, il codice id. A1, il drive 0, e il dispositivo 8.

Saranno utilizzati i valori di default per il disk drive (0) ed il numero di dispositivo (8) se non specificato diversamente. L'esempio che segue mostra un formato esteso del comando HEADER:

HEADER "DISCO", 51 RETURN

Il comando HEADER può essere inoltre utilizzato per cancellare tutti i dati da un disco usato, in modo che lo stesso disco possa essere utilizzato nuovamente come se fosse nuovo. Fare attenzione a non cancellare un disco il cui contenuto potrebbe risultare in seguito utile.

Può essere utilizzato il formato ridotto del comando HEADER se il disco è stato precedentemente formattato con il formato esteso del comando HEADER.

Il formato ridotto cancella l'elenco, cancellando tutti i dati allo stesso modo del formato esteso, ma mantiene lo stesso identificativo utilizzato precedentemente. Ecco come potrebbe apparire un formato ridotto del comando HEADER:

HEADER "NUOVIPROG" RETURN

Salvataggio su disco

In modalità C128 i programmi possono essere memorizzati su disco utilizzando uno dei due comandi seguenti:

DSAVE "NOME PROGRAMMA" RETURN

SAVE "NOME PROGRAMMA",8 RETURN

I due comandi sono equivalenti. Ricordare che la sequenza di carattere DSAVE" può essere visualizzata sullo schermo premendo il tasto funzione F5, oppure battendo la sequenza intera. Il nome di programma può essere composto da un massimo di 16 caratteri e dovrà essere racchiuso tra virgolette. Sullo stesso disco non possono essere presenti due programmi con lo stesso nome. Se si cerca di memorizzare un programma con lo stesso nome di un altro, il disco manterrà quello già memorizzato e non accetterà l'ultimo introdotto. Nel secondo esempio, 8 indica che si sta salvando il programma sul dispositivo numero 8. Con il comando DSAVE non è necessario specificare 8 in quanto il computer assume automaticamente che si sta utilizzando tale dispositivo.

Salvataggio su cassetta

Se si sta utilizzando un Datassette per la memorizzazione del programma, introdurre un nastro vuoto nel registratore, riavvolgere il nastro se necessario e battere:

SAVE "NOME PROGRAMMA" RETURN

Si deve battere la parola SAVE, seguita dal nome di programma, che può essere composto da un massimo di 16 caratteri.

NOTA: Lo schermo a 40 colonne si vuoterà nel corso del salvataggio del programma, e ritornerà allo stato normale al completamento dell'operazione.

Contrariamente a quanto avviene con i dischi, sulla cassetta si possono salvare due programmi con lo stesso nome. È consigliabile però non assegnare lo stesso nome a due programmi in quanto il computer caricherà sempre il primo programma presente sequenzialmente sul nastro.

Dopo il salvataggio, il programma potrà essere caricato nella memoria del computer ed eseguito ogni volta che lo si desidera.

Caricamento da disco

Il caricamento è semplicemente una copia del contenuto del programma dal disco alla memoria del computer. Se nella memoria è già presente un programma BASIC, al caricamento di un altro programma il primo verrà cancellato dalla memoria.

Per caricare un programma BASIC da disco, utilizzare uno dei due comandi seguenti in modalità C128:

DLOAD "NOME PROGRAMMA" RETURN

LOAD "NOME PROGRAMMA",8 RETURN

Ricordare che in modalità C128 è possibile utilizzare il tasto funzione F2 (premendo SHIFT e F1) per visualizzare la sequenza DLOAD", oppure è possibile battere la sequenza da tastiera. Nel secondo esempio, 8 indica al computer che si sta caricando dal dispositivo 8. Come DSAVE, DLOAD assume che il numero di dispositivo del disk drive sia 8. Battere il nome del programma esattamente come era stato salvato. In caso contrario il computer risponderà "FILE NOT FOUND".

Dopo aver caricato il programma, battere RUN per eseguirlo. Il Commodore 128 ha un formato speciale per il comando RUN utilizzato per caricare ed eseguire il programma in modalità C128

con un solo comando. Battere RUN seguito dal nome del programma (chiamato anche nome di file) racchiuso tra virgolette:

RUN "PROGRAMMA" RETURN

Caricamento da cassetta

Per caricare il programma da cassetta battere:

LOAD "NOME PROGRAMMA" RETURN

Se non si conosce il nome del programma, battere:

LOAD RETURN

Verrà così trovato il primo programma sul nastro. Nel corso della ricerca del programma da parte del Datassette lo schermo a 40 colonne è vuoto. Non appena il programma viene trovato, sullo schermo appare:

FOUND PROGRAM NAME

Per caricare il programma premere il tasto Commodore, oppure, in modalità C128, premere la barra spaziatrice per trovare il programma successivo sul nastro.

Per identificare il punto di inizio del programma utilizzare il contagiri sul Datassette. Quindi, per richiamare un programma, fare avanzare il nastro da 000 al punto di inizio del programma e battere:

LOAD RETURN

In questo caso non si dovrà specificare il NOME DI PROGRAMMA; il programma verrà caricato automaticamente in quanto si tratta del primo programma trovato sul nastro.

Altri comandi disco

Verifica di un programma

Per verificare il corretto salvataggio di un programma utilizzare il seguente comando in modalità C128.

DVERIFY "NOME PROGRAMMA" RETURN

Se il programma nel computer è identico a quello su disco, lo schermo visualizzerà "OK".

Il comando VERIFY funziona anche per i programmi su nastro. Battere:

VERIFY "NOME PROGRAMMA" RETURN

Non introdurre la virgola ed il numero di dispositivo.

Visualizzazione dell'elenco del disco

In modalità C128 è possibile visualizzare un elenco o directory dei programmi su disco utilizzando il seguente comando:

DIRECTORY RETURN

Con questa operazione viene listato il contenuto dell'elenco. Il modo più semplice per eseguire la stessa operazione è tramite la pressione del tasto funzione F3. Premendo F3 il C128 visualizzerà la parola "DIRECTORY" ed eseguirà il comando.

Per ulteriori dettagli sul salvataggio e il caricamento dei programmi, o per informazioni relative ai dischi, consultare il manuale del Datassette o del disk drive. Consultare inoltre le descrizioni dei comandi LOAD e SAVE nel Capitolo V, Enciclopedia BASIC 7.0.

****

In questa sezione del manuale sono state date le prime nozioni sul linguaggio BASIC ed alcuni concetti elementari di programmazione. La prossima sezione del manuale approfondisce questi concetti introducendo comandi, funzioni e tecniche addizionali da utilizzare nella programmazione in BASIC.

SEZIONE 4

Questa sezione descrive l'uso di molti comandi, funzioni e tecniche di programmazione BASIC che possono essere utilizzati in entrambe le modalità C128 e C64.

Questi comandi e funzioni permettono di programmare delle azioni ripetute attraverso tecniche di loop e di nidificazione; di gestire tabelle di valori; di spostarsi da una parte all'altra di un programma; di assegnare valori variabili ad una quantità e molte altre operazioni. Verranno dati esempi di programma per mostrare come questi concetti BASIC vengono applicati e come interagiscono.

Decisioni del computer - L'istruzione IF-THEN

Dopo aver appreso come modificare i valori delle variabili si vedrà ora come far prendere al computer le decisioni basate su questi valori aggiornati utilizzando l'istruzione IF-THEN. Si comunicherà al computer di eseguire un comando solo se (IF) si verifica una particolare condizione (es. IF X = 5). Il comando che il computer eseguirà quando la condizione viene soddisfatta si trova dopo la parola THEN nell'istruzione. Cancellare la memoria del computer battendo NEW seguito da RETURN, quindi battere il seguente programma:

```
10 J = 0
20 J=J+
30 ? J,"COMMODORE 128"
40 IF J = 5 THEN GOTO 60
50 GOTO 20
60 END
```

Ora non è più necessario premere il tasto STOP per arrestare un programma in loop. L'istruzione IF-THEN ordina al computer di continuare a stampare "COMMODORE 128" e di aumentare J finchè non si verifichi la condizione J = 5. Quando una condizione IF è falsa, il computer salta alla riga successiva di programma indipendentemente da ciò che si trova dopo la parola THEN.

Notare il comando END alla riga 60. È consigliabile introdurre sempre un'istruzione END come ultima riga di un programma per indicare al computer di fermare l'esecuzione delle istruzioni.

Di seguito è riportata una lista dei simboli che possono essere utilizzati nell'istruzione IF ed il loro significato:

SIMBOLO	SIGNIFICATO
=	**UGUALE**
>	**MAGGIORE DI**
<	**MINORE DI**
<>	**DIVERSO DA**
>=	**MAGGIORE DI O UGUALE A**
<=	**MINORE DI O UGUALE A**

Questi simboli operano seguendo regole matematiche. Vi sono diversi modi per determinare se una stringa è maggiore di, minore di o uguale ad un'altra. Consultare il Capitolo V, Enciclopedia BASIC 7.0 per maggiori informazioni sulle funzioni di "gestione stringa".

La Sezione 5 descrive alcuni potenti sviluppi del concetto IF-THEN, che consistono dei comandi BASIC 7.0 come BEGIN, BEND e ELSE.

Uso dei due punti

I due punti sono molto utili in fase di programmazione e vengono utilizzati per separare due (o più) comandi BASIC su una stessa riga.

Le istruzioni dopo un simbolo di due punti su una stessa riga vengono eseguite nell'ordine, da sinistra a destra. In una riga di programma possono essere inserite tante istruzioni quante possono essere contenute in un massimo di 160 caratteri, compreso il numero di riga. Questo equivale a 4 righe complete di schermo se in formato a 40 colonne e a due righe complete di schermo se in formato a 80 colonne. In questo modo si ha la possibilità di utilizzare appieno la parte THEN dell'istruzione IF-THEN. Infatti si potrà comunicare al computer di eseguire più comandi quando la condizione IF è vera. Cancellare la memoria del computer e battere il seguente programma:

```
10 N = 1
20 IF N<5 THEN PRINT N;"MINORE DI 5":GOTO 40
30 ? N; "MAGGIORE DI O UGUALE A 5"
40 END
```

Ora modificare la riga 10 in N = 40 ed eseguire nuovamente il programma. Notare che è possibile ordinare al computer di eseguire più di un'istruzione quando N è minore di 5. Dopo il comando THEN può essere introdotta una qualsiasi istruzione. Ricordare che GOTO 40 non sarà raggiunto finchè non si verificherà la condizione N<5. Ogni comando che dovrà essere eseguito **indipendentemente dal raggiungimento della condizione specificata** dovrà apparire su una riga separata.

Loop - Il comando FOR-NEXT

Nel programma utilizzato per la spiegazione di IF-THEN, si era ottenuta cinque volte la stampa di Commodore in quanto si era comunicato al computer di incrementare di uno la variabile J fino al raggiungimento del valore cinque, quindi era stata data l'istruzione di fine programma. In BASIC questa operazione può essere effettuata in modo più semplice, cioè utilizzando un loop FOR-NEXT, come il seguente:

```
10 FOR J = 1 T0 5
20 ? J,"COMMODORE 128"
30 NEXT J
40 END
```

Battere ed eseguire questo programma e confrontare il risultato con quello del programma IF-THEN. Il risultato è lo stesso. Infatti le fasi seguite dal computer sono quasi identiche per entrambi i programmi. Il loop FOR-NEXT è molto utile in fase di programmazione, infatti è possibile specificare quante volte il computer dovrà eseguire la stessa azione. Si esamineranno ora le fasi di esecuzione del programma precedente.

Per prima cosa il computer assegna alla variabile J il valore di 1. Il 5 nell'istruzione FOR nella riga 10 ordina al computer di eseguire tutte le istruzioni che si trovano tra l'istruzione FOR e l'istruzione NEXT fino a quando J non raggiungerà il valore di 5. In questo caso è presente una sola istruzione, l'istruzione PRINT.

Il computer assegna prima il valore 1 a J, quindi prosegue l'esecuzione del comando PRINT. Quando il computer raggiunge l'istruzione NEXT J, J viene incrementato e confrontato con 5. Se J non ha superato il valore di 5, il computer ritorna all'istruzionef PRINT. Dopo cinque esecuzioni di questo loop, il valore di J sarà 5, il programma salterà all'istruzione immediatamente successiva all'istruzione NEXT e proseguirà da quel punto. In questo esempio l'i-

struzione successiva è il comando END, quindi il programma si arresterà.

Loop vuoti - Inserimento di intervalli in un programma

Prima di procedere si vedrà come in alcuni casi i loop possono essere utilizzati per ritardare l'esecuzione di un programma che, fino a questo momento, il computer ha eseguito ad una velocità elevata. Cercare di capire come funzionerà il programma seguente prima di eseguirlo:

```
10 A$ = "COMMODORE C128"
20 FOR J = 1 TO 20
30 PRINT
40 FOR K = 1 TO 1500
50 NEXT K
60 PRINT A$
70 NEXT J
80 END
```

Il loop contenuto nelle righe 40 e 50 ha chiesto al computer di contare fino a 1500 prima di continuare l'esecuzione del programma. Questo viene chiamato loop di attesa e può essere molto utile. Trovandosi all'interno del loop principale del programma viene chiamato loop nidificato. Questo tipo di loop risulta utile quando si desidera che il computer esegua alcune operazioni in un certo ordine e ripeta l'intera sequenza di comandi un dato numero di volte.

La Sezione 5 descrive un modo più avanzato per inserire intervalli per mezzo del nuovo comando BASIC 7.0, SLEEP.

Il comando STEP

È possibile ordinare al computer di aumentare il contatore di una certa unità (ad es. 10, 0.5 o altra) utilizzando il comando STEP con l'istruzione FOR. Ad esempio, per contare fino a 100 di decina in decina, battere:

```
10 FOR X = 0 TO 100 STEP 10
20 ? X
30 NEXT
```

Se si esegue un solo loop alla volta non è necessario introdurre la X nell'istruzione NEXT - vedere più avanti in questa sezione. Notare che, oltre ad incrementare il contatore, è possibile decrementarlo. Ad esempio, modificare la riga 10 del programma prece-

dente come segue:

```
10 FOR X = 100 TO 0 STEP - 10
```

Il computer conterà a ritroso da 100 a 0, di decina in decina.

Non utilizzando un comando STEP con un'istruzione FOR il computer aumenterà il contatore di unità in unità.

I componenti del comando FOR-NEXT sono:

FOR - parola utilizzata per indicare l'inizio del loop.
X - variabile di conteggio; può essere utilizzata una qualsiasi variabile numerica.
1 - valore di inizio; può essere un numero qualsiasi, positivo o negativo.
TO - collega il valore iniziale a quello finale.
100 - valore finale; può essere un numero qualsiasi, positivo o negativo.
STEP - quando si utilizza un incremento diverso da 1.
2 - incremento; può essere un numero qualsiasi, positivo o negativo.

La Sezione 5 descrive un nuovo e potente comando BASIC 7.0, il comando DO/LOOP, che svolge una funzione simile a quella del comando STEP.

Introduzione di dati

Il comando INPUT

Assegnazione di un valore a una variabile

Cancellare la memoria del computer battendo NEW e premendo RETURN, quindi battere ed eseguire il seguente programma:

```
10 K = 10
20 FOR I = 1 TO K
30 ? "Commodore 128"
40 NEXT
```

In questo programma è possibile cambiare il valore di K nella riga 10 per eseguire il loop per il numero di volte desiderato. Questa operazione deve essere effettuata in fase di battitura del programma, prima dell'esecuzione. Di seguito verrà spiegato come comunicare al computer, dopo l'esecuzione del programma, quante volte eseguire lo stesso loop.

In altre parole, in questo esempio si desidera modificare il valore della variabile K ogni volta che il programma viene eseguito, senza però modificare il programma stesso. Questa viene chiamata possibilità di interagire con il computer. È infatti possibile,

utilizzando il comando INPUT, fare in modo che sia il computer stesso a chiedere quante volte deve essere eseguito un loop. Ad esempio, sostituire la riga 10 del programma con:

10 INPUT K

All'esecuzione del programma il computer risponderà con un ?. A questo punto occorre introdurre il valore da attribuire a K. Battere 15 e premere RETURN. Il computer eseguirà il loop 15 volte.

Messaggi guida (prompt)

È possibile fare in modo che il computer, attraverso un'istruzione INPUT, indichi la variabile da introdurre. Sostituire la riga 10 con:

10 INPUT "INTRODURRE UN VALORE PER K";K

Racchiudere tra virgolette il messaggio da stampare. Questo messaggio viene chiamato prompt. Utilizzare un punto e virgola tra le virgolette di chiusura del prompt e la lettera K. Nel prompt è possibile introdurre un qualsiasi messaggio, a patto che l'istruzione INPUT non superi i 160 caratteri di lunghezza, esattamente come avviene per tutti i comandi BASIC.

L'istruzione INPUT può essere utilizzata con le variabili stringa e alle stringhe vengono applicate le stesse regole delle variabili numeriche. Non dimenticare di utilizzare il simbolo **$** per identificare tutte le variabili stringa. Cancellare la memoria del computer battendo NEW e premendo RETURN, quindi introdurre il programma seguente:

10 INPUT "COME TI CHIAMI";N$
20 ? "CIAO"; N$

Ora eseguire il programma con il comando RUN. Alla visualizzazione di "COME TI CHIAMI", battere il proprio nome. Non dimenticare di premere RETURN dopo il nome.

Dopo l'introduzione del valore di una variabile (numerica o stringa) per mezzo di INPUT si potrà fare riferimento a tale variabile in qualunque punto del programma usando il nome della variabile stessa. Battere ?N**$** <RETURN>. Il computer ricorderà il nome.

Il comando GET

Per interagire con il computer possono essere utilizzati nel programma altri comandi BASIC. Uno di questi è il comando GET ed è simile ad INPUT. Per capire il funzionamento del comando GET, cancellare la memoria del computer e battere il seguente programma:

```
10 GET A$
20 IF A$ = "" THEN 10
30 ? A$
40 END
```

Quando si batte RUN seguito da RETURN sembra che il computer non abbia alcuna reazione. Questo succede perchè il computer attende la pressione di un tasto. Il comando GET, in effetti, dice al computer di controllare la tastiera e di trovare il carattere o il tasto che è stato premuto. La richiesta del programma viene soddisfatta anche non battendo alcun carattere, grazie alla riga 20. Infatti questa riga ordina al computer di tornare alla riga 10 in caso non venga battuto alcun carattere (il carattere nullo è indicato da due serie di virgolette senza spazio tra di loro). Il loop continuerà fino alla pressione di un tasto. Il computer assegna quindi a A$ il carattere del tasto premuto.

Il comando GET è molto utile per programmare un tasto sulla tastiera. Nell'esempio precedente premendo Q viene visualizzato un messaggio sullo schermo. Battere ed eseguire il programma, quindi premere Q.

```
10 ? "PREMERE Q PER VISUALIZZARE IL MESSAGGIO"
20 GET A$
30 IF A$ = "" THEN 20
40 IF A$ = "Q" THEN 60
50 GOTO 20
60 FOR I = 1 TO 25
70 ? "ORA POSSO USARE L'ISTRUZIONE GET"
80 NEXT
90 END
```

Notare che premendo un tasto diverso da Q, il computer non visualizzerà il messaggio, ma tornerà alla riga 20 in attesa di un altro carattere.

La Sezione 5 descrive come utilizzare l'istruzione GETKEY, un nuovo e più potente comando BASIC 7.0 che può essere usato per l'esecuzione di quanto sopra.

Esempio di programma

Per applicare il loop FOR-NEXT ed il comando INPUT, cancellare la memoria del computer battendo NEW RETURN, quindi introdurre il programma seguente:

```
 10 T = 0
 20 INPUT "QUANTI NUMERI";N
 30 FOR J = 1 TO N
 40 INPUT "INTRODURRE UN NUMERO";X
 50 T = T + X
 60 NEXT
 70 A = T/N
 80 PRINT
 90 ? "VI SONO";N"NUMERI PER UN TOTALE DI";T
100 ? "MEDIA =";A
110 END
```

Con questo programma è possibile indicare di quanti numeri si vuole la media. I numeri possono essere cambiati ogni volta che viene eseguito il programma senza dover modificare il programma stesso.

Verifichiamo ora le operazioni svolte dal computer riga per riga:

La riga 10 assegna a T il valore di 0 (somma dei numeri).
La riga 20 permette di determinare di quanti numeri, contenuti nella variabile N, ottenere la media.
La riga 30 dice al computer di eseguire il loop N volte.
La riga 40 permette di introdurre i numeri effettivi di cui si desidera la media.
La riga 50 somma ogni numero al valore totale.
La riga 60 dice al computer di aumentare il contatore (J) e di tornare alla riga 30 mentre il contatore (J) è <=N.
La riga 70 divide il totale per la somma dei numeri battuti (N) dopo l'esecuzione del loop per N volte.
La riga 80 visualizza una riga vuota.
La riga 90 stampa il messaggio indicante quanti numeri sono stati introdotti e la loro somma.
La riga 100 stampa la media dei numeri.
La riga 110 comunica al computer che il programma è finito.

Il comando READ-DATA

Esiste un altro metodo per comunicare al computer il numero di caratteri da utilizzare in un programma: l'istruzione READ, usata all'interno del programma per permettere al computer di rilevare un certo numero di caratteri dall'istruzione DATA. Ad esempio, per trovare la media di cinque numeri, utilizzare le istruzioni READ e DATA nel modo seguente:

```
10 T = 0
20 FOR J = 1 TO 5
30 READ X
40 T = T + X
50 NEXT
60 A = T/5
70 ? "MEDIA=";A
80 END
90 DATA 5,12,1,34,18
```

All'esecuzione del programma il computer stampa MEDIA = 14. Il programma utilizza la variabile T per mantenere aggiornato il totale e calcola la media esattamente come il programma dell'esempio precedente. Il programma READ-DATA però trova i numeri da cui ottenere una media in una riga DATA. Notare la riga 30, READ X. Il comando READ indica al computer la presenza di un'istruzione DATA nel programma. Il computer trova la riga DATA ed utilizza il primo numero come valore corrente per la variabile X. Quando il loop viene eseguito la seconda volta, come valore per X sarà utilizzato il secondo numero nell'istruzione DATA e così via.

Nell'istruzione DATA è possibile inserire un numero qualsiasi ma non un calcolo. L'istruzione DATA può trovarsi in qualsiasi punto del programma, anche dopo l'istruzione END. Questo perchè il computer non esegue mai realmente l'istruzione DATA, ma fa solo riferimento ad essa. Separare sempre con virgole i diversi elementi, ma non introdurre mai una virgola tra la parola DATA ed il primo numero della lista.

Se nel programma sono presenti più istruzioni DATA, il computer farà riferimento a quella più vicina all'istruzione READ in corso di esecuzione. Il computer contrassegna con un puntatore l'ultimo dato che è stato letto. Dopo la lettura da parte del computer del primo numero nell'istruzione DATA, il puntatore si sposterà sul numero successivo. Quando il computer ritorna sull'istruzione

READ, assegna al nome di variabile nell'istruzione READ il valore che il puntatore sta indicando.

All'interno di un programma possono essere utilizzate tante istruzioni READ e DATA quante sono necessarie, assicurarsi però che nelle istruzioni DATA siano presenti dati sufficienti rispetto all'istruzione READ relativa. Togliere dall'istruzione DATA nell'ultimo programma uno dei numeri ed eseguire il programma. Il computer visualizzerà ?OUT OF DATA ERROR IN 30. In questo caso si è verificato che, alla esecuzione del loop per la quinta volta, il computer non ha trovato alcun dato da leggere. Questo è il significato del messaggio di errore. Al contrario, l'introduzione di troppi dati nell'istruzione DATA non causa problemi al computer in quanto il computer stesso non rileva l'esistenza dei dati superflui.

Il comando RESTORE

Il comando RESTORE può essere utilizzato all'interno di un programma per reimpostare sul primo dato il puntatore dei dati. Nel programma precedente, sostituire l'istruzione END (riga 80) con quanto segue:

```
80 RESTORE
```

e aggiungere:

```
85 GOTO 10
```

Ora eseguire il programma. Il programma continuerà ad essere eseguito utlizzando la stessa istruzione DATA. NOTA: Se il computer visualizza il messaggio OUT OF DATA ERROR, è perchè ci si è dimenticati di inserire un nuovo numero al posto di quello cancellato precedentemente nell'istruzione DATA, quindi tutti i dati vengono esauriti prima che l'istruzione READ venga eseguita il numero di volte richiesto.

È possibile utilizzare le istruzioni DATA per assegnare un valore alle variabili stringa. Lo stesso avviene per i dati numerici. Cancellare la memoria del computer e battere il seguente programma:

```
10 FOR J = 1 TO 3
20 READ A$
30 ? A$
40 NEXT
50 END
60 DATA COMMODORE,128,COMPUTER
```

Se l'istruzione READ richiede una variabile stringa, nell'istruzione DATA possono essere inserite lettere o numeri. Notare comunque che, siccome il computer sta leggendo una stringa, i numeri saranno memorizzati come una stringa di caratteri e non come un valore con la possibilità di essere modificato. I numeri memorizzati come stringhe possono essere stampati ma non utilizzati in calcoli. Inoltre non è possibile introdurre lettere in un'istruzione DATA se l'istruzione READ richiede una variabile numerica.

Uso delle matrici

Si è visto fino ad ora come utilizzare READ-DATA per fornire molti valori ad una variabile. Di seguito verrà spiegato come memorizzare tutti i dati in un'istruzione DATA invece di sostituire il valore di una variabile con nuovi dati o come ad esempio richiamare il terzo numero o la seconda stringa di caratteri.

Ogni volta che viene assegnato un nuovo valore ad una variabile, il computer cancella il valore precedente nella cella di memoria della variabile e memorizza al suo posto il nuovo valore. È possibile comunicare al computer di riservare una fila di celle di memoria e di memorizzare ogni valore assegnato a quella variabile nel programma. Questa fila di celle viene chiamata matrice.

Variabili indicizzate

Se la matrice contiene tutti i valori assegnati alla variabile X nell'esempio READ-DATA, viene chiamata matrice X. Il primo valore assegnato a X nel programma viene chiamato X(1), il secondo X(2) ecc. Queste vengono chiamate variabili indicizzate. I numeri tra parentesi vengono chiamati indici ed è possibile utilizzare una variabile od un calcolo come indice. Di seguito viene data una seconda versione del programma della media utilizzando questa volta le variabili indicizzate.

```
 5 DIM X(5)
10 T = 0
20 FOR J = 1 TO 5
30 READ X(J)
40 T= T + X(J)
50 NEXT
60 A = T/5
70 ? "MEDIA =";A
80 END
90 DATA 5,12,1,34,18
```

Notare che non sono state effettuate molte modifiche. L'unica istruzione nuova è la riga 5, che dice al computer di riservare cinque celle di memoria per la matrice X. La riga 30 è stata modificata in modo tale che il computer, ogni volta che viene eseguito il loop, assegni alla posizione nella matrice X un valore dall'istruzione DATA che corrisponda al contatore di loop (J). La riga 40 calcola il totale, esattamente come nell'esempio precedente dello stesso programma, utilizzando però una variabile indicizzata.

Dopo l'esecuzione del programma, per richiamare ad esempio il terzo numero, battere ?X(3)<RETURN>. Il computer ricorda ogni numero della matrice X. Allo stesso modo si possono creare matrici stringa per memorizzare i caratteri nelle variabili stringa. Cercare di aggiornare il programma READ-DATA COMMODORE 128 COMPUTER in modo che il computer possa ricordare gli elementi nella matrice A$.

```
 5 DIM A$(3)
10 FOR J = 1 TO 3
20 READ A$(J)
30 ? A$(J)
40 NEXT
50 END
60 DATA COMMODORE,C128, COMPUTER
```

CONSIGLIO: Nel programma non è necessaria l'istruzione DIM a meno che la matrice utlizzata comprenda più di 10 elementi. Vedere DIMENSIONAMENTO DELLE MATRICI.

Dimensionamento delle matrici

Le matrici possono essere utilizzate con loop nidificati permettendo al computer di gestire i dati in modo più sofisticato. Supponiamo ora di avere una tabella di 10 righe, con cinque numeri su ogni riga e di voler trovare la media dei cinque numeri di ogni riga. Si potrebbero creare 10 matrici e dire al computer di calcolare la media dei cinque numeri di ogni riga. Questo però non è necessario in quanto si possono comprendere i numeri in una matrice bidimensionale. Questa matrice avrà le stesse dimensioni della tabella di numeri su cui lavorare - 10 righe per 5 colonne. L'istruzione DIM per questa matrice (che chiameremo matrice X) sarà:

```
10 DIM X(10,5)
```

Con questo si richiede al computer di riservare spazio in memoria per una matrice bidimensionale denominata X. Il computer assegnerà spazio sufficiente per 50 numeri. Non è necessario riempire una matrice con tutti i numeri per cui è stata dimensionata, il computer però riserva spazio sufficiente per ognuna delle posizioni della matrice.

Esempio di programma

Ora risulta semplice fare riferimento ad un qualsiasi numero della tabella indicando la colonna e la riga. Vedere il diagramma seguente. Trovare il terzo elemento nella decima riga (1500). Nel programma si chiamerà questo numero X(10,3). Il programma riportato di seguito legge i numeri della tabella in una matrice bidimensionale (X) e calcola la media (average) dei numeri in ciascuna riga (row).

	Colonna				
RIGA	**1**	**2**	**3**	**4**	**5**
1	**1**	**3**	**5**	**7**	**9**
2	**2**	**4**	**6**	**8**	**10**
3	**5**	**10**	**15**	**20**	**25**
4	**10**	**20**	**30**	**40**	**50**
5	**20**	**40**	**60**	**80**	**100**
6	**30**	**60**	**90**	**120**	**150**
7	**40**	**80**	**120**	**160**	**200**
8	**50**	**100**	**150**	**200**	**250**
9	**100**	**200**	**300**	**400**	**500**
10	**500**	**1000**	**1500**	**2000**	**2500**

```
10 DIM X(10,5), A(10)
20 FOR R=1 TO 10
30 T=0
40 FOR C=1 TO 5
50 READ X(R,C)
60 T=T+X(R,C)
70 NEXT C
80 A(R)=T/5
90 NEXT R
100 FOR R=1 TO 10
110 PRINT "ROW #";R
120 FOR C=1 TO 5
130 PRINT X(R,C)
140 NEXT C
150 PRINT "AVERAGE =";A(R)
160 FOR D=1 TO 1000:NEXT
170 NEXT R
180 DATA 1,3,5,7,9
190 DATA 2,4,6,8,10
200 DATA 5,10,15,20,25
210 DATA 10,20,30,40,50
220 DATA 20,40,60,80,100
230 DATA 30,60,90,120,150
240 DATA 40,80,120,160,200
250 DATA 50,100,150,200,250
260 DATA 100,200,300,400,500
270 DATA 500,1000,1500,2000,2500
280 END
```

Sottoprogrammi

Il comando GOSUB-RETURN

Fino ad ora, l'unico metodo utilizzato per comunicare al computer di spostarsi su un'altra parte del programma è stato tramite l'uso del comando GOTO. Di seguito verrà spiegato come saltare a un'altra parte del programma, eseguire le istruzioni di quella parte, e quindi ritornare al punto di partenza e continuare l'esecuzione del programma.

La parte di programma su cui il computer si sposta e che viene eseguita viene denominata **sottoprogramma** o **subroutine**. Cancellare la memoria del computer e battere il seguente programma:

```
 10 A$ = "SUBROUTINE":B$="PROGRAMMA"
 20 FOR J = 1 TO 5
 30 INPUT "INTRODURRE UN NUMERO";X
 40 GOSUB 100
 50 PRINT B$:PRINT
 60 NEXT
 70 END
100 PRINT A$:PRINT
110 Z = X↑ 2:PRINT Z
120 RETURN
```

Questo programma eleva al quadrato i numeri introdotti e stampa il risultato. Gli altri messaggi di stampa indicano l'esecuzione da parte del computer della subroutine o del programma principale. La riga 40 indica al computer di spostarsi sulla riga 100, di eseguirla e di eseguire le istruzioni successive a tale riga fino al ritrovamento di un comando RETURN. L'istruzione RETURN dice al computer di tornare alla riga immediatamente successiva al comando GOSUB e di continuare l'esecuzione. Il sottoprogramma può trovarsi in qualsiasi punto del programma - anche dopo l'istruzione END. Inoltre, ricordare che i comandi GOSUB e RETURN devono essere sempre utilizzati in combinazione tra loro (come FOR-NEXT e IF-THEN), in caso contrario il computer visualizza un messaggio di errore.

Il comando ON GOTO/GOSUB

Esiste un altro metodo, chiamato salto condizionato, per permettere al computer di saltare ad un'altra sezione del programma: utilizzando l'istruzione ON. Con questo metodo il computer de-

cide a quale parte di un programma saltare basandosi su un calcolo o un'introduzione da tastiera. L'istruzione ON viene utilizzata con il comando GOTO o il comando GOSUB-RETURN, a seconda delle esigenze. Dopo il comando ON dovrà essere introdotta una variabile o un calcolo. Dopo il comando GOTO o GOSUB dovrà essere presente una lista di numeri di riga. Battere il programma seguente per capire il funzionamento del comando ON.

```
 10 ? "INTRODURRE UN NUMERO COMPRESO TRA UNO
    E CINQUE"
 20 INPUT X
 30 ON X GOSUB 100,200,300,400,500
 40 END
100 ? "IL NUMERO ERA UNO":RETURN
200 ? "IL NUMERO ERA DUE":RETURN
300 ? "IL NUMERO ERA TRE":RETURN
400 ? "IL NUMERO ERA QUATTRO":RETURN
500 ? "IL NUMERO ERA CINQUE":RETURN
```

Quando il valore di X è 1, il computer salta al primo numero di riga della lista (100). Quando X è 2, il computer salta al secondo numero della lista (200) e così via.

Uso delle locazioni di memoria

Uso di PEEK e POKE per l'accesso a RAM o ROM

Ogni area di memoria del computer ha una funzione speciale. Ad esempio, è prevista un'ampia area per la memorizzazione di programmi e delle variabili ad essi associate. Questa parte di memoria, chiamata RAM viene cancellata utilizzando il comando NEW. Le altre aree non sono così ampie, ma possiedono funzioni specifiche. Ad esempio, esiste un'area di locazioni di memoria che controlla le funzioni musicali del computer.

Per l'accesso e la gestione della memoria del computer vengono utilizzati due comandi BASIC: PEEK e POKE. L'uso dei comandi PEEK e POKE può risultare molto utile in fase di programmazione in quanto il contenuto delle locazioni di memoria del computer determina esattamente le azioni che il computer intraprenderà in un particolare momento.

Uso di PEEK

PEEK può essere utilizzato per conoscere un valore che il computer sta memorizzando in una locazione di memoria (una locazione di memoria che può contenere un valore da 0 a 255). È possibile prelevare il valore di una qualsiasi locazione di memoria (RAM o ROM) con il comando PEEK in modalità DIRETTA o PROGRAMMA. Battere:

P = PEEK(2594) RETURN

? P RETURN

Alla pressione di RETURN dopo la prima riga, il computer assegna alla variabile P il valore della locazione di memoria 2594. Quindi stampa il valore alla pressione di RETURN dopo l'introduzione del comando ? P. La locazione di memoria 2594 determina se i tasti come la barra spaziatrice e CRSR funzioneranno ripetutamente se tenuti premuti. Un valore 128 nella locazione 2594 provoca questa azione. Tenere premuta la barra spaziatrice ed osservare come il cursore si muove sullo schermo.

Uso di POKE

Per modificare il valore memorizzato in una locazione RAM, utilizzare il comando POKE. Battere:

POKE 2594,96 RETURN

Il computer memorizza il valore che si trova dopo la virgola (96) nella locazione di memoria prima della virgola (2594). Un valore 96 nella locazione di memoria 2594 indica al computer di non rendere ripetitivi i tasti come la barra spaziatrice e CRSR quando sono tenuti premuti. Ora tenere premuta la barra spaziatrice ed osservare il cursore. Il cursore si muoverà di un solo spazio verso destra. Per riportare il computer allo stato normale, battere:

POKE 2594,128 RETURN

Non è possibile alterare il valore di tutte le locazioni di memoria; infatti i valori nella ROM possono essere letti ma non modificati.

NOTA: Questi esempi assumono che ci si trovi nel banco di memoria 0. Vedere la descrizione del comando BANK nel Capitolo V, Enciclopedia BASIC 7.0 per ulteriori dettagli sui banchi.

Funzioni principali

Che cos'è una funzione

Una funzione è un'operazione predefinita del linguaggio BASIC che in genere fornisce un valore singolo. Quando la funzione fornisce il valore si dice che "ritorna" il valore. Ad esempio, la funzione SQR (elevare al quadrato) è una funzione matematica che ritorna il valore di un numero specifico quando questo viene elevato al quadrato.

Esistono due tipi di funzioni:

Numeriche - ritornano un risultato in forma di numero singolo. Le funzioni numeriche vanno dal calcolo di valori matematici alla specifica del valore numerico di una locazione di memoria.

Stringa - ritorna un risultato in forma di carattere.

Di seguito vengono descritte alcune delle funzioni più comunemente utilizzate. Vedere il Capitolo V, Enciclopedia BASIC 7.0 per una lista completa delle funzioni BASIC 7.0.

La funzione INTEGER (INT)

Per arrotondare un numero per difetto occorre utilizzare la funzione INT, che toglie qualsiasi valore che si trovi dopo un punto decimale. Battere gli esempi seguenti:

? INT(4.25) RETURN

? INT(4.75) RETURN

? INT(SQR(50)) RETURN

Se si desidera arrotondare per eccesso, considerare il secondo esempio che dà come risultato 5. Per arrotondare un numero con un decimale superiore a 0,5, aggiunger 0,5 al numero prima di utilizzare la funzione INT. In questo modo i numeri con decimali superiori a 0,5 saranno aumentati di 1 prima di essere arrotondati per difetto dalla funzione INT. Battere:

? INT(4.75 + 0.5) RETURN

Il computer aggiunge 0,5 a 4,75 prima dell'esecuzione della funzione INT in modo da ottenere 5,25 che sarà arrotondato a 5. Per vedere come arrotondare il risultato di un'operazione, battere:

? INT((100/6) + 0.5) RETURN

È possibile sostituire con qualsiasi operazione la divisione contenuta nelle parentesi interne.
Per arrotondare i numeri allo 0,01 più vicino, invece di aggiungere 0,5 al numero, occorrerà aggiungere 0,005 e moltiplicare per 100. Ad esempio, per arrotondare 2,876 allo 0,01 più vicino, battere:

?(2.876 = 0.005)*100 RETURN

Utilizzare ora la funzione INT per cancellare tutto quello che si trova dopo il punto decimale (che si era spostato di due posizioni verso destra dopo la moltiplicazione per 100). L'operazione sarà:

? INT((2.876 = 0.005)*100) RETURN

e darà un totale di 288. Dividere ora per 100 per ottenere il valore di 2,88, che è la risposta corretta. Con questa tecnica è possibile arrotondare allo 0,01 superiore calcoli come il seguente:

? INT+((2.876 + 1.29 + 16.1–9.534) + 0.005)
***100/100 RETURN**

Generazione di numeri casuali - La funzione RND

La funzione RND dice al computer di generare un numero casuale. Questo può risultare utile nella simulazione di giochi d'azzardo e nella creazione di programmi grafici o musicali. Tutti i numeri casuali sono di nove cifre, in forma decimale, da 0,000000001 a 0.999999999. Battere:

? RND (0) RETURN

Moltiplicando per sei il numero generato casualmente la gamma dei numeri generati raggiunge valori da maggiore di 0 a minore di 6. Per includere il numero 6 tra i numeri generati, occorre aggiungere uno al risultato di RND(0)*6, ottenendo 1<X<7. Utilizzando la funzione INT per eliminare i decimali, il comando genererà numeri interi da 1 a 6. Questo processo può essere utilizzato nella simulazione di un dado che rotola. Battere:

```
10 R = (INT(RND(1)*6 + 1)
20 ? R
30 GOTO 10
```

Ogni numero generato rappresenta il lancio di un solo dado. Per simulare due dadi, utilizzare due comandi di questo tipo. Ogni numero viene generato separatamente e la somma dei due numeri rappresenta il totale dei due dadi.

Le funzioni ASC E CHR$

Ogni carattere (compresi i caratteri grafici) che può essere visualizzato dal Commodore 128 corrisponde a un codice numerico. Questo numero viene denominato codice di stringa di carattere (CHR$) e il Commodore 128 ne comprende 255. A questo concetto sono associate due funzioni. La prima è la funzione ASC. Battere:

?ASC("Q") RETURN

Il computer visualizza 81, che è il codice di stringa carattere del tasto Q. Sostituire la Q nel comando dell'esempio precedente con un qualsiasi altro carattere per scoprire il numero di codice ASCII Commodore per ogni singolo carattere.

La seconda funzione è la funzione CHR$. Battere:

?CHR$(81) RETURN

Il computer risponde Q. La funzione CHR$ è infatti l'opposto della funzione ASC ed entrambe fanno riferimento alla tabella dei codici di stringa di carattere nella memoria del computer. I valori CHR$ possono essere utilizzati per la programmazione dei tasti funzione. Per ulteriori dettagli sull'uso di CHR$, consultare la Sezione 5. Vedere l'Appendice E di questo manuale per avere un elenco completo dei codici ASC e CHR$.

Conversione di stringhe e numeri

A volte può essere necessario eseguire dei calcoli sui caratteri numerici memorizzati come variabili di stringa in un programma o eseguire operazioni di stringa su numeri. Per questo motivo esistono due funzioni BASIC che possono essere utilizzate per convertire le variabili da numeriche a stringa e viceversa.

La funzione VAL

La funzione VAL ritorna un valore numerico per un argomento di stringa. Cancellare la memoria del computer e battere il programma seguente:

```
10 A$ = "64"
20 A = VAL(A$)
30 ? "IL VALORE DI";A$;"È";A
40 END
```

La funzione STR$

La funzione STR$ ritorna la rappresentazione di stringa di un valore numerico. Cancellare la memoria del computer e battere il seguente programma:

10 A = 65
20 A$ = STR$(A)
30 ? A"È IL VALORE DI";A$

La funzione radice quadrata (SQR)

Utilizzare la funzione SQR per ottenere la funzione radice quadrata. Ad esempio, per trovare la radice quadrata di 50, battere:

? SQR(50) RETURN

In questo modo è possibile trovare la radice quadrata di qualsiasi numero.

La funzione valore assoluto (ABS)

La funzione valore assoluto (ABS) è molto utile quando si lavora con numeri negativi. Questa funzione viene utilizzata per ottenere il valore positivo di un numero qualsiasi, positivo o negativo. Battere gli esempi seguenti:

? ABS(– 10) RETURN

? ABS(5)"È UGUALE A "ABS(–5) RETURN

I comandi STOP e CONT

È possibile arrestare l'esecuzione di un programma e riprenderla dal punto in cui era stata interrotta includendo nel programma il comando STOP. L'istruzione STOP può essere introdotta in qualsiasi punto del programma. Quando il computer si arresta (cioè interrompe l'esecuzione del programma), è possibile utilizzare i comandi in modalità DIRETTA per visualizzare le varie fasi del programma. Ad esempio si può trovare il valore di un contatore di loop o altre variabili. Questa caratteristica risulta molto utile in fase di messa a punto del programma. Cancellare la memoria del computer e battere il programma seguente:

```
10 X = INT(SQR(630))
20 Y = (.025*80)↑2
30 Z = INT(X*Y)
40 STOP
50 FOR J = 0 TO Z STEP Y
60 ? "ARRESTA E CONTINUA"
70 NEXT
80 END
```

Ora eseguire il programma. Il computer visualizza "BREAK IN 40". A questo punto il computer ha calcolato i valori di X, Y e Z. Per visualizzare come procederà il programma, battere: PRINT X;Y;Z. Spesso può essere utile, in fase di messa a punto di un programma lungo (o uno corto ma complicato), sapere il valore di una variabile ad un certo punto nel corso dell'esecuzione del programma.

Per riprendere l'esecuzione del programma, battere CONT e premere RETURN se non si sono effettuate modifiche nel programma. Il computer riprende l'esecuzione dall'istruzione che si trova subito dopo il comando STOP.

****

Questa sezione del manuale e la sezione precedente sono state studiate per permettere all'utente di familiarizzare con il linguaggio di programmazione BASIC e con le sue caratteristiche. Le sezioni successive di questo capitolo descrivono i comandi propri della modalità Commodore 128. Alcuni comandi di modalità Commodore 128 forniscono prestazioni che non sono però disponibili in modalità C64. Altri comandi della modalità Commodore 128 funzionano come alcuni comandi C64, ma in modo più semplice. La sintassi per tutti i comandi Commodore 7.0 viene fornita nel Capitolo V, Enciclopedia BASIC 7.0.

SEZIONE 5

Introduzione

Questa sezione introduce all'uso di alcuni potenti comandi e istruzioni BASIC che probabilmente anche i programmatori BASIC esperti non conoscono. Programmando in linguaggio BASIC si saranno incontrate sicuramente situazioni in cui questi comandi e istruzioni avrebbero potuto essere di aiuto. Questa sezione spiega i concetti utilizzati per la creazione di questi comandi e dà esempi sull'uso di ciascuno di questi comandi in un programma (consultare il Capitolo V, Enciclopedia BASIC 7.0 per avere una lista completa e la spiegazione di questi comandi ed istruzioni). Questa sezione spiega inoltre come utilizzare i tasti speciali disponibili in modalità C128.

Uso avanzato del loop

L'istruzione DO/LOOP

L'istruzione DO/LOOP offre modi più sofisticati di creazione di un loop rispetto alle istruzioni GOTO, GOSUB e FOR/NEXT. La combinazione DO/LOOP offre al linguaggio BASIC una tecnica potente e versatile di solito disponibile solo per linguaggi di programmazione strutturati. In questa parte del manuale vengono illustrati alcuni utilizzi di DO/LOOP.

Per creare un loop infinito, iniziare con un'istruzione DO, quindi introdurre la riga o le righe che specificano l'azione che il computer dovrà intraprendere. Quindi completare con un'istruzione LOOP, come nell'esempio seguente:

```
100 DO
110 PRINT "RIPETIZIONE"
120 LOOP
```

Premere il tasto **RUN/STOP** per arrestare il programma.

Le azioni specificate dopo l'istruzione DO vengono eseguite fino al raggiungimento dell'istruzione LOOP (riga 120); quindi il controllo viene trasferito nuovamente all'istruzione DO (riga 100). Perciò, tutte le istruzioni che si trovano tra DO e LOOP verranno eseguite all'infinito.

Until

Un'altra tecnica molto utile è la combinazione dell'istruzione UNTIL con DO/LOOP. L'istruzione UNTIL imposta una condizione perchè il loop possa essere eseguito. Il loop continua ad essere eseguito finchè non si verifica la condizione UNTIL.

```
100 DO:INPUT "TI PIACE IL TUO COMPUTER";A$
110 LOOP UNTIL A$ = "SÌ"
120 PRINT "GRAZIE"
```

L'istruzione DO/LOOP viene spesso utilizzata per ripetere indefinitamente nel corpo di un programma un'intera routine, come avviene nell'esempio seguente:

```
10 PRINT "IL PROGRAMMA CONTINUA FINCHÈ NON SI
   BATTE 'QUIT'"
20 DO UNTIL A$ = "QUIT"
30 INPUT "GRADI FAHRENHEIT";F
40 C = (5/9)*(F – 32)
50 PRINT F;"GRADI FAHRENHEIT UGUALI A";C;
   "GRADI CELSIUS"
60 INPUT "CONTINUARE O ABBANDONARE";A$
70 LOOP
80 END
```

L'istruzione DO/LOOP può anche essere utilizzata come contatore, dove l'istruzione UNTIL viene utilizzata per specificare un determinato numero di ripetizioni.

```
10 N = 2*2
20 PRINT "DUE PER DUE UGUALE";N
30 DO UNTIL X = 25
40 X = X + 1
50 N = N*2
60 PRINT"PER DUE";X + 1;"PER...";N
70 LOOP
80 END
```

Notare che, tralasciando l'istruzione contatore (UNTIL X = 25 nella riga 30), il numero verrà raddoppiato all'infinito finchè non sarà incontrato un errore di OVERFLOW.

While

L'istruzione WHILE funziona allo stesso modo di UNTIL, ma il loop viene ripetuto solo mentre è effettiva la condizione, come nell'esempio seguente:

```
100 DO:INPUT "TI PIACE IL TUO COMPUTER";A$
110 LOOP WHILE A$ <> "SÌ"
120 PRINT "GRAZIE"
```

Exit

L'istruzione EXIT può essere utilizzata all'interno di DO/LOOP. Quando viene incontrata l'istruzione EXIT il programma salta all'istruzione successiva all'istruzione LOOP.

La clausola ELSE con IF-THEN

La clausola ELSE fornisce un modo per dire al computer come rispondere in caso non si verifichi la condizione di IF-THEN. Invece di continuare l'esecuzione della riga di programma successiva, il computer eseguirà il comando o salterà alla riga di programma menzionata nella clausola ELSE. Ad esempio, per stampare l'elevazione al quadrato di un numero, utilizzare la clausola ELSE come segue:

```
10 INPUT "BATTERE UN NUMERO DA ELEVARE AL
   QUADRATO";N
20 IF N< 100 THEN PRINT N*N: ELSE 40
30 END
40 ?"IL NUMERO DEVE ESSERE < 100": GOTO 10
```

Tra l'istruzione IF-THEN e la clausola ELSE deve essere inserito un simbolo di due punti.

La sequenza BEGIN/BEND con IF-THEN

Il BASIC 7.0 permette di ampliare le possibilità di IF-THEN. La sequenza BEGIN/BEND permette di includere delle righe di programma che saranno eseguite se si verifica la condizione IF, invece di eseguire una semplice azione o un GOTO. Il comando è strutturato come segue:

```
IF condizione THEN BEGIN:
   (righe di programma):
   BEND:ELSE
```

Assicurarsi di aver battuto un simbolo di due punti tra BEGIN e qualsiasi altra istruzione e tra l'ultimo comando della sequenza e la parola BEND. BEGIN/BEND può essere utilizzato senza una clausola ELSE oppure può essere utilizzato seguendo la clausola ELSE se THEN è seguito da un solo comando. Battere il programma seguente:

```
10 INPUT A
20 IF A< 100 THEN BEGIN:? "IL NUMERO ERA";A
30 SLEEP 2:REM PAUSA
40 FOR X = 1 TO A
50 ?"QUESTO È UN ESEMPIO DI BEGIN/BEND"
60 NEXT X
70 ?"VA BENE":BEND:ELSE?"TROPPO GRANDE"
80 END
```

Questo programma chiede di introdurre un numero. Se il numero è minore di 100, vengono eseguite le istruzioni tra le parole chiave BEGIN e BEND, comprese tutte le istruzioni che si trovano sulla stessa riga di BEND (tranne ELSE). Sullo schermo appare il messaggio "IL NUMERO ERA N". La riga 30 è un loop di ritardo per mantenere il messaggio sullo schermo per un tempo sufficiente perchè possa essere letto facilmente. Quindi viene utilizzato un loop FOR/NEXT per visualizzare un messaggio per il numero di volte indicato dall'utente. Se il numero è maggiore di 100, la condizione THEN non viene presa in considerazione e viene eseguita la condizione ELSE (che stampa "TROPPO GRANDE"). La parola chiave ELSE deve trovarsi sulla stessa riga di BEND.

Il comando SLEEP

Notare l'uso del comando SLEEP nella riga 30 del programma qui sopra. Con il comando SLEEP è possibile introdurre in modo più semplice e preciso una pausa del numero di secondi voluti in un'operazione di programma. Il formato del comando SLEEP è:

SLEEP n

dove n indica il numero di secondi, da 0 a 65535, di ritardo desiderati. Nel comando illustrato nella riga 30, il numero 2 specifica una pausa di due secondi.

Formattazione dell'output

Il comando PRINT USING

Supponiamo di dover scrivere un programma di vendite che calcoli un valore in dollari. Le vendite totali divise per il numero di venditori dà una media delle vendite. Però l'esecuzione di questo calcolo potrebbe dare dei risultati con quattro o cinque cifre decimali. Per avere solamente due cifre decimali è possibile formattare il risultato che il computer stampa utilizzando il comando PRINT USING.

PRINT USING permette di creare un formato per l'output, utilizzando spazi, virgole, punti decimali e simboli di dollaro. I simboli di numero (#) vengono utilizzati per rappresentare spazi o caratteri nel risultato visualizzato. Ad esempio:

PRINT USING "#$######.##";A

indica al computer che quando viene stampato A, questo dovrà essere nel formato dato, con un massimo di sei cifre alla sinistra del punto decimale e due cifre decimali dopo il punto. Il simbolo di numero davanti al simbolo di dollaro indica che il $ sarà mobile, cioè dovrà sempre essere posto vicino all'ultimo numero a sinistra nel formato.

Per visualizzare una virgola per indicare le migliaia, come in $1,000.00, aggiungere la virgola nell'istruzione PRINT USING. Gli output possono infatti essere formattati includendo virgole, punti decimali e simboli di dollaro. Per maggiori dettagli sull'uso di altri caratteri speciali per PRINT USING, consultare l'Enciclopedia BASIC.

Il comando PUDEF

Se si desidera un output formattato che rappresenti altri simboli oltre a dollari e centesimi, utilizzare il comando PUDEF. I quattro caratteri di formato possono essere sostituiti da un carattere qualsiasi sulla tastiera.

Il comando PUDEF ha quattro posizioni, ma non è necessario ridefinirle tutte. Il comando appare come segue:

```
PUDEF"   ,.$"
        1234
```

Dove:

- la posizione 1 è il carattere riempitore. Se questa posizione non viene ridefinita, appare uno spazio vuoto.
- la posizione 2 è il carattere virgola (default).
- la posizione 3 è il punto decimale.
- la posizione 4 è il simbolo di dollaro.

Ad esempio, per scrivere un programma che converta i totali da dollari in sterline, formattare l'output utilizzando i comandi seguenti:

```
10 PUDEF ",.£"
20 PRINT USING "#$####.##";X
```

Programma di esempio

Questo programma calcola gli interessi e i pagamenti di un prestito utilizzando alcuni dei comandi e delle istruzioni viste precedentemente. Imposta un valore minimo per il prestito utilizzando la clausola ELSE con un'istruzione IF-THEN ed imposta un formato dollari e centesimi con PRINT USING.

```
10 INPUT "PRESTITO IN DOLLARI";A
20 IF A< 100 THEN 70: ELSE P = .15
30 I = A*P
40 ?"TOTALE DEL PAGAMENTO UGUALE";
50 PRINT USING "#$#####.##";A + I
60 GOTO 80
70 ?"PRESTITI INFERIORI A $100 NON DISPONIBILI"
80 END
```

Introduzione di dati con il comando GETKEY

Si è visto fino ad ora come utilizzare i comandi INPUT e GET per introdurre i dati nel corso di un programma. Un altro modo per introdurre dati in fase di esecuzione del programma è tramite l'istruzione GETKEY. L'istruzione GETKEY accetta un solo tasto alla volta. GETKEY è seguito generalmente da una variabile stringa (ad esempio A$) alla quale viene assegnato il tasto premuto. GETKEY è utile in quanto permette di introdurre i dati carattere per carattere senza dover premere RETURN dopo ciascun carattere. L'istruzione GETKEY può essere utilizzata esclusivamente all'interno di un programma.

Di seguito viene dato un esempio sull'utilizzo di GETKEY in un programma:

```
1000 PRINT "SCEGLIERE A, B, C, D, E OPPURE F"
1010 GETKEY A$
1020 PRINT A$;"È IL TASTO PREMUTO."
```

Il computer attende la pressione di un singolo tasto, a questo punto il carattere viene assegnato alla variabile A$ e visualizzato nella riga 1020. Il programma seguente mostra un uso più complesso e più utile di GETKEY: per scegliere una delle risposte possibili ad un'unica domanda e anche per chiedere se la domanda deve essere ripetuta. Se la risposta data non è corretta, l'utente potrà riprovare premendo il tasto "S" (riga 80). Il tasto premuto per la risposta, viene assegnato alla variabile A$, mentre la risposta "RIPROVATE" viene assegnata a B$ attraverso le istruzioni

GETKEY nelle righe 60 e 90. Le istruzioni IF/THEN vengono utilizzate per i loop nel programma per avere una reazione appropriata da parte del computer a seconda dei diversi input da tastiera.

```
 10 PRINT "CHI HA SCRITTO 'IL CORVO'?"
 20 PRINT "A. EDGAR ELLEN POE"
 30 PRINT "B. EDGAR ALLAN POE"
 40 PRINT "C. IGOR ALLEN POE"
 50 PRINT "D. ROB CORVO"
 60 GETKEY A$
 70 IF A$ = "B" THEN 150
 80 PRINT "SBAGLIATO. RIPROVATE? (S oppure N)"
 90 GETKEY B$
100 IF B$ = "S" THEN PRINT "A,B,C OPPURE D?":GOTO 60
110 IF B$ = "N" THEN 140
120 PRINT "BATTERE S OPPURE N - RIPROVATE"
130 GOTO 90
140 PRINT "LA RISPOSTA CORRETTA È B."
145 GOTO 160
150 PRINT "ESATTO"
160 END
```

GETKEY è molto simile a GET tranne per il fatto che GETKEY attende automaticamente la pressione di un tasto.

Consigli utili per la programmazione

Nelle sezioni precedenti si è visto come effettuare delle modifiche nei programmi e come correggere gli errori di battitura con l'aiuto di INST/DEL. Oltre a questo, il BASIC fornisce altri comandi e funzioni che permettono di localizzare errori di programma e comandi che facilitano le operazioni di programmazione.

Introduzione di programmi

Auto

IL BASIC C128 fornisce un metodo di auto-numerazione con il quale è possibile determinare l'incremento dei numeri di riga. Ad esempio, per numerare il programma nel modo normale, con numeri di dieci in dieci in modalità DIRETTA, battere:

AUTO 10 RETURN

Il computer numererà automaticamente le righe di programma di dieci in dieci. Alla pressione di RETURN, apparirà il numero di riga successivo, con il cursore sul punto in cui dovrà essere introdotta la prossima istruzione. È possibile scegliere una numerazione con un incremento a piacere, ad esempio di 5 in 5 o di 50 in 50, ecc. Per questa operazione è sufficiente introdurre il numero desiderato dopo la parola AUTO e premere RETURN. Per disattivare l'auto-numerazione, battere AUTO senza però introdurre alcun numero e premere RETURN.

Renumber

Se si aggiungono istruzioni a un programma, la numerazione delle righe potrebbe risultare confusa. Con il comando RENUMBER si potranno modificare i numeri di riga con incremento costante sia per l'intero programma, sia per parte di esso. Il comando RENUMBER possiede diversi parametri opzionali, come mostrato di seguito:

RENUMBER [[nuova riga iniziale][,[incremento]
incremento][,vecchia riga iniziale]]]]]

La nuova riga iniziale sarà numerata come lo sarebbe la riga di programma dopo aver dato il comando RENUMBER. Se non viene specificato altrimenti, il valore di default sarà 10. L'incremento è il valore tra un numero di riga e l'altro ed anch'esso ha come default 10. Il vecchio numero di riga iniziale è il numero di riga da cui dovrà iniziare la rinumerazione. Questa caratteristica permette di rinumerare una sola parte di un programma senza doverlo rinumerare per intero. Per default la rinumerazione inizierà dalla prima riga del programma. Ad esempio:

RENUMBER 40,,80

ordina al computer di rinumerare il programma a partire dalla riga 80, con incrementi di 10 in 10. La riga 80 diventa la riga 40.

Notare che questo comando, come il comando AUTO, può essere eseguito solo in modalità diretta.

Delete

Si è visto in precedenza come cancellare righe di programma battendo il numero di riga seguito da RETURN. Questa operazione potrebbe risultare lunga quando si desiderano cancellare più righe in un programma. Il comando DELETE permette di rispar-

miare tempo in quanto si possono specificare più righe che verranno cancellate in una sola volta. Ad esempio:

DELETE 10 – 50

cancellerà le righe da 10 a 50. L'uso di DELETE è simile all'uso di LIST; infatti con questo comando è possibile specificare fino a quale riga o a partire da quale riga si dovrà cancellare oppure la singola riga da cancellare, come mostrato negli esempi seguenti:

DELETE–120
cancellerà tutte le righe fino alla 120 compresa

DELETE 120–
cancellerà la riga 120 e tutte le righe seguenti

DELETE 120
cancellerà solo la riga 120

Identificazione di errori all'interno di un programma

Quando un programma non funziona come dovrebbe, generalmente viene visualizzato un messaggio di errore. A volte però i messaggi sono vaghi e non aiutano nell'identificazione dell'errore. Con il computer Commodore 128 vi sono diversi modi per poter localizzare un problema.

Help

Il Commodore 128 mette a disposizione un comando HELP che specifica la riga in cui si è verificato un errore. Per attivare il comando HELP, premere il tasto speciale HELP sulla fila di tasti situati al di sopra della tastiera principale.

Battere l'istruzione seguente con l'errore intenzionale in essa contenuto:

10 ?3;4:5;6

All'esecuzione di questo programma di una riga, il computer stampa 3 e 4 seguiti però dal messaggio "SYNTAX ERROR IN 10". Supponiamo che non ci si sia resi conto dell'errore (un simbolo di due punti al posto di un punto e virgola tra il 4 e il 5). Premere il tasto HELP (oppure battere la parola HELP e premere RETURN). Il computer visualizza nuovamente la riga, evidenziando 5;6 per indicare che l'errore è stato riscontrato in quella riga.

Individuazione di errori - Il comando TRAP

Normalmente, quando in un programma viene riscontrato un errore, il programma si arresta. A quel punto, è possibile premere il tasto HELP per rintracciare l'errore, oppure utilizzare il comando BASIC 7.0 TRAP. Con questo comando viene introdotta all'interno del programma una funzione di intercettazione errori. Il comando TRAP localizza un errore e comunica di correggerlo, quindi riprende il programma dal punto interrotto. Normalmente, la funzione di individuazione errori viene introdotta come prima riga di un programma.

```
5 TRAP 100
```

Questa istruzione comunica al computer di saltare ad una determinata riga di programma (in questo caso alla riga 100) quando viene riscontrato un errore. La riga 100 appare alla fine del programma e imposta una condizione. Questa riga non viene eseguita se non vengono riscontrati errori. Al rilevamento di un errore, l'istruzione TRAP viene eseguita ed il controllo viene diretto ad un'altra parte del programma. Queste istruzioni possono essere utilizzate per localizzare eventuali errori in fase di introduzione dati, per riprendere l'esecuzione del programma oppure per passare dalla modalità grafica alla modalità testo e così via. Se si esegue l'esempio con DO/LOOP (che raddoppiava i numeri) senza introdurre un'istruzione UNTIL, viene visualizzato un errore OVERFLOW ed il programma si arresta. Per prevenire quanto sopra è sufficiente aggiungere due righe, una all'inizio e una alla fine del programma come ad esempio:

```
  5 TRAP 100
100 IF N>1 THEN END
```

Sebbene il valore di N sia stato sempre molto maggiore di uno per l'intero programma, l'istruzione non viene considerata finchè non viene riscontrato un errore. Quando il numero è troppo grande, cioè supera la capacità del computer, l'istruzione TRAP entra in funzione. Siccome N è maggiore di uno, il programma viene concluso (anzichè interrompersi).

Di seguito viene dato un esempio in cui l'intercettazione viene visualizzata per prevenire che lo zero venga sottoposto a divisione:

```
  10 TRAP 1000
 100 INPUT "POSSO ESEGUIRE LA DIVISIONE DI
     QUALSIASI NUMERO. INTRODUCI UN NUMERO DA
     DIVIDERE";D
 110 INPUT "PER QUALE NUMERO LO DEVO
     DIVIDERE";B
 120 A = D/B
 130 PRINT D;"DIVISO PER";B;"UGUALE";A
 140 END
1000 IF B = 0 THEN PRINT "NON POSSO ESEGUIRE
     QUESTA OPERAZIONE"
1100 INPUT "SCEGLI UN ALTRO NUMERO";B;RESUME 120
```

Notare l'istruzione RESUME nella riga 1100 per comunicare al computer di tornare alla riga menzionata (in questo caso la 120) e di continuare. A seconda dell'errore localizzato la ripresa potrà essere o non essere possibile.

Per ulteriori informazioni sull'intercettazione di errori, vedere le funzioni di errore ERR$, EL e ER, descritte nel Capitolo V. Enciclopedia BASIC 7.0.

Tracciamento di un programma - I comandi TRON e TROFF

Quando si incontra un problema con un programma, oppure se non si ottengono i risultati attesi, potrebbe risultare utile eseguire metodicamente il programma compiendo esattamente le stesse azioni che compierebbe il computer. Questo processo viene chiamato tracciamento. Disegnare dei riquadri variabili ed aggiornare i valori a seconda delle istruzioni di programma. Eseguire il calcolo e stampare il risultato seguendo ogni istruzione.

Il tracciamento potrebbe rivelare ad esempio che si è utilizzato un comando GOTO con un numero di riga errato oppure che si è calcolato un risultato ma non lo si è memorizzato in una variabile. Molti errori di programma possono essere riscontrati fingendo di essere noi stessi il computer ed eseguendo una sola istruzione per volta. Il C128 può eseguire una specie di analisi per mezzo dei comandi TRON e TROFF. In fase di esecuzione del programma, il computer per mezzo del comando TRON stampa i numeri di riga nell'ordine in cui le righe vengono eseguite. In questo modo è possibile verificare il motivo per cui il programma non dà i risultati attesi.

Battere uno qualsiasi dei programmi eseguiti fino a questo momento oppure uno di propria creazione. Per attivare la modalità di tracciamento, battere TRON in modalità DIRETTA. All'esecuzione del programma, notare come i numeri di riga appaiano tra parentesi prima della visualizzazione dei risultati. Cercare di seguire i numeri di riga e di scoprire in quante fasi il computer è arrivato ad un determinato punto. TRON sarà molto più interessante lavorando su programmi con più "salti", come ad esempio righe contenenti GOTO, GOSUB e IF-THEN. Prima di continuare battere TROFF per disattivare la modalità di tracciamento.

Non è necessario tracciare un programma per intero; è possibile infatti introdurre TRON in un programma come riga immediatamente precedente la sezione di programma che causa il problema e TROFF come riga di programma alla fine di tale sezione. All'esecuzione del programma saranno racchiuse tra parentesi nei risultati solo le righe tra TRON e TROFF.

Creazione di finestre

Le finestre sono un'area specifica dello schermo che l'utente definisce come spazio di lavoro. Tutto quello che viene battuto (righe, listati di programmi, ecc.) dopo l'impostazione della finestra appare all'interno del perimetro della finestra stessa, senza uscirne al di fuori. Il Commodore 128 fornisce due metodi di creazione delle finestre: il comando WINDOW e le funzioni del tasto ESC.

Uso del comando WINDOW per la creazione di una finestra

Il linguaggio BASIC 7.0 del Commodore 128 mette a disposizione un comando che permette di creare e gestire finestre: il comando WINDOW. Il formato di questo comando è:

WINDOW colonna in alto a sinistra, riga in alto a sinistra, colonna in basso a destra, riga in basso a destra [,opzione di cancellazione]

I primi due numeri dopo WINDOW specificano il numero della colonna e della riga che formeranno l'**angolo in alto a sinistra** della finestra; i due numeri seguenti sono le coordinate dell'**angolo in basso a destra**. Ricordare che queste coordinate devono essere in accordo al tipo di formato dello schermo (40 o 80 colonne). È inoltre possibile includere nel comando un'opzione di cancellazione. Aggiungendo 1 alla fine del comando, l'area della finestra sullo schermo verrà cancellata, come nell'esempio seguente:

WINDOW 10, 10, 20, 20, 1

Nell'esempio di programma seguente, sullo schermo verranno create quattro finestre sia in formato a 40 colonne sia a 80 colonne.

```
10 SCNCLR:COLOR 5,5         :REM CLEAR SCREEN & SELECT PURPLE TEXT.
20 COLOR5,5
30 WINDOW0,0,39,24          :REM SET WINDOW TO FULL SCREEN
40 COLOR 0,13:COLOR 4,13 :REM SET 40 SCREEN TO MED GREY.
50 A$="ABCDEFGHIJKLMNOPQRST"
60 COLOR 5,5                :REM SELECT PURPLE TEXT.
70 FORI=1 TO 25             :REM FILL SCREEN WITH CHARACTERS.
80 PRINT A$;A$:NEXT I
90 WINDOW 1,1,7,20          :REM DEFINE WINDOW 1.
100 COLOR 5,3               :REM PRINT A$ IN REVERSE RED TEXT.
110 PRINT CHR$(18);A$;
120 WINDOW 15,15,39,20,1 :REM DEFINE SECOND WINDOW.
130 COLOR 5,7               :REM PRINT A$ IN CYAN TEXT.
140 FOR I=1 TO 6:PRINT A$;:NEXT I
150 WINDOW 30,1,39,22,1  :REM DEFINE THIRD WINDOW.
160 COLOR 5,8:LIST          :REM LIST IN YELLOW TEXT.
170 WINDOW 5,5,33,18,1   :REM DEFINE FOURTH WINDOW.
180 COLOR 5,2               :REM PRINT A$ AND LIST IN WHITE TEXT.
190 PRINT A$:LIST
200 END
```

Uso del tasto ESC per la creazione di una finestra

Per impostare una finestra con il tasto ESC, seguire le fasi elencate qui sotto:

1. Portare il cursore sul punto dello schermo che sarà l'angolo superiore sinistro della finestra.
2. Premere e rilasciare il tasto ESC, quindi premere T.
3. Portare il cursore sul punto dello schermo che sarà l'angolo inferiore destro della finestra.
4. Premere e rilasciare ESC, quindi premere B. La finestra è così impostata.

Il tasto ESC può essere utilizzato per modificare la finestra ed il testo in essa contenuto. Le funzioni di modifica schermo, come quelle di introduzione e cancellazione testo, scorrimento e modifica delle dimensioni della finestra, possono essere eseguite premendo ESC seguito da un altro tasto. Per usare una funzione specifica, premere e rilasciare ESC, quindi premere uno dei tasti elen-

cati di seguito:

- **@** Cancella tutto quello che si trova a partire dalla posizione del cursore fino alla fine della finestra
- **A** Modalità inserimento automatico
- **B** Imposta l'angolo inferiore destro della finestra (alla posizione corrente del cursore)
- **C** Annulla la modalità di inserimento automatico
- **D** Cancella la riga corrente
- **E** Imposta il cursore sulla modalità di non-lampeggiamento
- **F** Imposta il cursore sulla modalità di lampeggiamento
- **G** Attiva la segnalazione acustica (tasti Control-G)
- **H** Disattiva la segnalazione acustica
- **I** Inserisce una riga
- **J** Porta all'inizio della riga corrente
- **K** Porta alla fine della riga corrente
- **L** Attiva lo scorrimento dello schermo
- **M** Disattiva lo scorrimento dello schermo
- **N** Riporta alla visualizzazione normale (non inversione schermo)(solo per modalità a 80 colonne)
- **O** Disattiva le modalità di inserimento e "quote"
- **P** Cancella tutto quello che si trova a partire dall'inizio della riga fino alla posizione del cursore
- **Q** Cancella tutto quello che si trova a partire dalla posizione del cursore fino alla fine della riga
- **R** Inverte la visualizzazione sullo schermo (solo per la modalità a 80 colonne)
- **S** Trasforma il cursore in un quadratino (■)
- **T** Imposta l'angolo superiore sinistro della finestra (alla posizione corrente del cursore)
- **U** Trasforma il cursore in un trattino (–)
- **V** Fa scorrere lo schermo verso l'alto di una riga
- **W** Fa scorrere lo schermo verso il basso di una riga
- **X** Passa da 40 a 80 colonne
- **Y** Ripristina i punti di tabulazione di default
- **Z** Cancella i punti di tabulazione.

Provare ad utilizzare le funzioni del tasto ESC. Probabilmente si troveranno alcune funzioni più utili di altre. Notare che è possibile utilizzare anche il normale tasto INST/DEL per eseguire modifiche di testo all'interno di una finestra.

Dopo l'impostazione della finestra, tutto lo schermo sarà racchiuso nel riquadro che è stato definito. Per cancellare l'area della finestra, premere SHIFT contemporaneamente a CLEAR/HOME. Per cancellare la finestra, premere due volte CLEAR/HOME. La finestra sarà così cancellata ed il cursore sarà posizionato nell'angolo superiore sinistro dello schermo. Le finestre risultano particolarmente utili quando si scrivono, listano ed eseguono i programmi in quanto permettono di lavorare in una particolare area dello schermo lasciando inalterata la parte rimanente dello schermo.

Funzionamento a 2 MHz

I comandi FAST e SLOW

La modalità operativa 2 MHz permette di eseguire programmi non grafici in formato a 80 colonne ad una velocità doppia di quella normale. Per passare dalla modalità normale a quella veloce utilizzare i comandi FAST e SLOW.

Il comando FAST riporta il Commodore 128 alla modalità a 2 MHz. Il formato del comando è:

FAST

Il comando SLOW riporta il Commodore 128 alla modalità a 1 MHz. Il formato del comando è:

SLOW

Tasti utilizzabili solo in modalità C128

Tasti funzione

I quattro tasti sul lato destro della tastiera del Commodore 128, al di sopra del tastierino numerico sono tasti funzione speciali che permettono di risparmiare tempo nell'esecuzione dei compiti ripetitivi utilizzando un solo tasto. Il primo tasto è contrassegnato **F1/F2**, il secondo **F3/F4**, il terzo **F5/F6** e l'ultimo **F7/F8**. Per utilizzare le funzioni 1, 3, 5 e 7, premere i tasti da soli, mentre per utilizzare i tasti funzione 2,4,6 e 8, premere **SHIFT** contemporaneamente al tasto funzione.

Di seguito vengono elencate le funzioni standard di ogni tasto:

F1	F2	F3	F4
GRAPHIC	**DLOAD"**	**DIRECTORY**	**SCNCLR**
F5	F6	F7	F8
DSAVE"	**RUN**	**LIST**	**MONITOR**

Ecco una spiegazione delle funzioni di ciascun tasto:

TASTO 1 attiva una delle modalità grafiche quando viene introdotto il numero dell'area grafica e viene premuto RETURN. Il comando GRAPHIC è necessario per impartire i comandi grafici come ad esempio CIRCLE o PAINT. Per ulteriori dettagli consultare la Sezione 6.

TASTO 2 stampa DLOAD" sullo schermo. Per caricare un programma da disco, invece di battere DLOAD per intero, è sufficiente introdurre il nome del programma, e premere RETURN.

TASTO 3 lista un elenco dei file sul disco che si trova nel disk drive.

TASTO 4 cancella lo schermo usando il comando SCNCLR.

TASTO 5 stampa DSAVE" sullo schermo. Per salvare il programma corrente sul disco, è sufficiente introdurre il nome del programma e premere RETURN.

TASTO 6 esegue il programma corrente.

TASTO 7 visualizza un listato del programma corrente.

TASTO 8 permette di accedere al Monitor del Linguaggio Macchina. Vedere l'Appendice J per una descrizione del Monitor.

Ridefinizione dei tasti funzione

È possibile ridefinire o programmare uno qualsiasi di questi tasti per eseguire la funzione desiderata. L'operazione di ridefinizione è semplice se si utilizza il comando KEY. Si possono infatti ridefinire i tasti dai programmi BASIC oppure modificarli in qualsiasi momento in modalità diretta. Un esempio di quando potrebbe essere utile la ridefinizione è quando si utilizza spesso un comando e si desidera memorizzarlo per evitare di doverlo battere ogni volta per intero. Le nuove definizioni vengono annullate allo spegnimento del computer. Si possono ridefinire tanti tasti quanti se

ne desidera e per quante volte è necessario.

Per riprogrammare il tasto funzione F7 per passare alla modalità testo dalle modalità ad alta risoluzione o grafica multicolor, ad esempio, si userà un comando come il seguente:

KEY 7,"GRAPHIC 0" + CHR$(13)

CHR$(13) è il carattere in codice ASCII di RETURN. Quindi premendo F7 dopo aver ridefinito il tasto, si otterrà la scrittura automatica del comando "GRAPHIC 0" e la sua introduzione nel computer per mezzo di RETURN. Ad un solo tasto possono essere assegnati interi comandi o serie di comandi.

Altri tasti usati solo nella modalità C128

Help

Come si sarà notato precedentemente, quando si commette un errore in un programma, il computer visualizza un messaggio di errore. Questi messaggi di errore sono spiegati in dettaglio nell'Appendice A di questo manuale. Per ottenere aiuto per quanto riguarda gli errori, utilizzare il tasto HELP. Dopo la visualizzazione di un messaggio di errore, premere il tasto HELP per localizzare il punto esatto in cui è stato commesso l'errore.
Alla pressione di HELP, la riga contenente l'errore verrà evidenziata sullo schermo con inversione video (per il formato a 40 colonne) oppure verrà sottolineata (per il formato a 80 colonne). Ad esempio:

?SYNTAX ERROR IN LINE 10 Messaggio sullo schermo

HELP Premere HELP.

10 PRONT "COMMODORE COMPUTER"
La riga contenente l'errore viene evidenziata con inversione video se in formato a 40 colonne, oppure viene sottolineata se in formato a 80 colonne.

No Scroll

Premere questo tasto per arrestare lo scorrimento del testo quando il cursore raggiunge la parte inferiore dello schermo. Questo disattiva lo scorrimento fino a quando viene premuto nuovamente il tasto NO SCROLL.

Caps Lock

Con questo tasto è possibile battere tutte le lettere maiuscole senza dover utilizzare il tasto SHIFT. Il tasto CAPS LOCK attiva le maiuscole quando viene premuto e le disattiva quando viene premuto una seconda volta. Questo tasto ha effetto solo sui tasti alfabetici.

Visualizzazione a 40/80 colonne

Il tasto **40/80** seleziona il formato per lo schermo principale (di default): a 40 o 80 colonne. Lo schermo selezionato visualizza tutti i messaggi e gli output al momento dell'accensione o quando vengono premuti i tasti RESET o RUN/STOP/RESTORE. Questo tasto può essere usato per impostare il formato di visualizzazione **solo prima** dell'accensione o alla reinizializzazione del computer. Con questo tasto non è possibile cambiare modalità **dopo** l'accensione del computer, a meno che non si utilizzino i tasti RUN/STOP/RESET o RUN/STOP/RESTORE. La Sezione 8 fornisce spiegazioni dettagliate sulle modalità a 40/80 colonne.

Alt

Il tasto ALT permette l'assegnazione da parte dei programmi di una funzione speciale ad alcuni tasti o serie di tasti.

Quando il tasto ALT viene premuto insieme a qualche altro tasto non si ottiene alcun effetto, a meno che questo tasto non sia stato ridefinito tramite un programma applicativo specifico.

Tab

Questo tasto funziona come il tasto di tabulazione di una macchina per scrivere e può essere usato per impostare o annullare i punti di tabulazione sullo schermo e per spostare il cursore alla colonna su cui è impostata la tabulazione.

Line Feed

Questo tasto permette di portare il cursore sulla riga successiva, come avviene utilizzando il tasto di movimento cursore verso il basso.

****

Questa sezione spiega solo alcuni concetti, tasti e comandi che fanno del Commodore 128 una macchina speciale. Per ulteriori spiegazioni sul linguaggio BASIC consultare l'Enciclopedia BASIC 7.0 nel Capitolo V.

SEZIONE 6

Grafica

In modalità C128, il linguaggio BASIC 7.0 del Commodore 128 fornisce nuovi e potenti comandi e istruzioni per facilitare la programmazione di grafici. Entrambi i due formati di schermo disponibili in modalità C128 (a 40 e 80 colonne) vengono controllati da un chip di microprocessore separato. Il chip per le 40 colonne viene denominato controllore dell'interfaccia video, o **VIC** (dall'inglese Video Interface Controller). Il chip per le 80 colonne viene chiamato **8563**. Il chip **vic** mette a disposizione 16 colori e permette l'esecuzione della grafica ad alta precisione chiamata grafica a matrice di punti. Il chip per le 80 colonne lavora anch'esso con 16 colori ma visualizza solo caratteri e caratteri grafici. Per cui tutti i programmi per lo sviluppo di grafici ad alta precisione in modalità C128 dovranno essere eseguiti nel formato a 40 colonne.

Caratteristiche grafiche

Il Commodore 128 fornisce le seguenti caratteristiche grafiche in modalità C128:

- 13 comandi di grafica speciali
- 16 colori
- Sei differenti modalità di visualizzazione
- Otto figure mobili programmabili chiamate SPRITE
- Visualizzazione combinata grafica/testo

Tutte queste caratteristiche permettono di avere un sistema grafico versatile e semplice da usare.

Riassunto dei comandi

Di seguito viene data una breve spiegazione di ogni comando grafico:

BOX	- Disegna rettangoli sullo schermo a matrice di punti
CHAR	- Visualizza i caratteri sullo schermo a matrice di punti
CIRCLE	- Disegna cerchi, ellissi e altre figure geometriche
COLOR	- Seleziona i colori della cornice, del primo piano, dello sfondo e dei caratteri dello schermo
DRAW	- Visualizza righe e punti sullo schermo a matrice di punti
GRAPHIC	- Seleziona una visualizzazione (testo, matrice di punti o matrice di punti a divisione di schermo)
PAINT	- Colora l'area sullo schermo a matrici di punti
SCALE	- Imposta la dimensione relativa delle immagini sullo schermo a matrice di punti
SPRDEF	- Attiva la modalità di definizione sprite per modificare la forma degli sprite
SPRITE	- Attiva, colora ed imposta le priorità di schermo degli sprite ed espande gli sprite
SPRSAV	- Memorizza una variabile stringa di testo in un'area di memorizzazione sprite e viceversa
SSHAPE	- Memorizza l'immagine di una parte dello schermo a matrice di punti in una variabile stringa di testo

La maggior parte di questi comandi è descritta negli esempi di questa sezione. Per ulteriori informazioni sui formati e le funzioni dei comandi grafici, compresi quelli non trattati in questa sezione, consultare il Capitolo V, Enciclopedia BASIC 7.0.

Programmazione grafica con il C128

La sezione seguente descrive, fase per fase, un esempio di programmazione grafica. Nel corso dell'apprendimento di ogni comando grafico, consigliamo di aggiungerlo ad un programma che verrà costruito mano a mano che si procede in questa sezione. Alla fine della sezione, si sarà creato un programma grafico completo.

Scelta dei colori

La prima fase di una sessione di programmazione grafica è la scelta dei colori per lo sfondo, per il primo piano e per la cornice dello schermo. Per selezionare i colori, battere:

COLOR sorgente, colore

dove **sorgente** è la sezione dello schermo che si sta colorando (sfondo, primo piano, cornice, ecc.) e **colore** è il codice colore della sorgente. Vedere la Figura 6-1 per riferimenti per i numeri sorgente, la Figura 6-2 per i numeri di colore nel formato a 40 colonne e la Figura 6-3 per i numeri di colore nel formato a 80 colonne.

Numero	Sorgente
0	Colore di sfondo a 40 colonne (VIC)
1	Primo piano dello schermo grafico (VIC).
2	Colore 1 del primo piano per lo schermo multicolore (VIC)
3	Colore 2 del primo piano per lo schermo multicolore (VIC)
4	Cornice a 40 colonne (VIC) (in modalità testo o grafica)
5	Colore carattere dello schermo di testo a 40 o 80 colonne
6	Colore di sfondo a 80 colonne (8563)

Figura 6-1. Numeri sorgente

Codice Colore	Colore	Codice Colore	Colore
1	Nero	9	Arancio
2	Bianco	10	Marrone
3	Rosso	11	Rosso chiaro
4	Azzurro	12	Grigio scuro
5	Porpora	13	Grigio
6	Verde	14	Verde chiaro
7	Blu	15	Blu chiaro
8	Giallo	16	Grigio chiaro

Figura 6-2. Numeri di colore nel formato a 40 colonne

Codice Colore	Colore	Codice Colore	Colore
1	Nero	9	Porpora scuro
2	Bianco	10	Marrone
3	Rosso scuro	11	Rosso chiaro
4	Azzurro chiaro	12	Azzurro scuro
5	Porpora chiaro	13	Grigio
6	Verde scuro	14	Verde chiaro
7	Blu scuro	15	Blu chiaro
8	Giallo chiaro	16	Grigio chiaro

Figura 6-3. Numeri di colore nel formato a 80 colonne

Tipi di visualizzazione di schermo

Il C128 fornisce diversi tipi di visualizzazione delle informazioni sullo schermo; il parametro "sorgente" del comando COLOR si riferisce alle diverse modalità di visualizzazione. I tipi di visualizzazione su schermo vengono suddivisi in tre categorie.

La prima categoria è quella di visualizzazione del testo, che visualizza solamente caratteri come lettere, numeri, simboli speciali ed i caratteri grafici rappresentati sul lato frontale di quasi tutti i tasti del C128. Il C128 visualizza il testo sia nel formato a 40 colonne sia in quello a 80 colonne.

La seconda categoria di visualizzazione viene utilizzata per i grafici ad alta precisione, come illustrazioni e disegni complicati. Questo tipo di modalità di visualizzazione comprende la modalità standard a matrice di punti e la modalità multicolore a matrice di punti. Le modalità a matrice di punti permettono di controllare ciascun punto dello schermo o **pixel** (elemento dell'immagine). Questo permette di ottenere un'estrema precisione nelle illustrazioni e in altri lavori grafici eseguiti con il computer formato a 80 colonne è specifico per la visualizzazione del testo.

La differenza tra la modalità testo e la modalità a matrice di punti è nel modo in cui ogni schermo invia e memorizza le informazioni. Lo schermo di testo può gestire caratteri interi, ognuno dei quali copre un'area di 8 per 8 pixel sullo schermo. La più potente modalità a matrice di punti invece controlla ogni singolo pixel dello schermo.

Il terzo tipo di visualizzazione, lo schermo diviso, è un insieme delle prime due modalità. La visualizzazione a schermo diviso visualizza parte dello schermo come testo e parte in modalità a matrice di punti (standard o multicolore). Questa caratteristica è possibile in quanto il C128 usa due parti separate della memoria del computer per la memorizzazione dei due schermi: una parte destinata al testo ed una parte alla grafica.

Battere il seguente breve programma:

```
10 COLOR 0,1:REM COLORE DI SFONDO TESTO = NERO
20 COLOR 1,3:REM COLORE PRIMO PIANO SCHERMO A
   MATRICE DI PUNTI = ROSSO
30 COLOR 4,1:REM COLORE CORNICE = NERO
```

Questo esempio colora di nero lo sfondo, di rosso il primo piano e di nero la cornice.

Selezione della modalità grafica

La prossima fase di programmazione grafica è la selezione della modalità grafica appropriata. Questo viene eseguito utilizzando il comando GRAPHIC il cui formato è il seguente:

GRAPHIC <modalità [,c][,s]/CLR>

dove **modalità** è un numero da 0 a 5, c è 0 o 1 e s è un valore da 0 a 25. La Figura 6-4 mostra i valori corrispondenti alle modalità grafiche.

Modalità	Descrizione
0	Testo standard a 40 colonne
1	Matrice di punti standard
2	Matrice di punti standard (schermo diviso)
3	Matrice di punti multicolore
4	Matrice di punti multicolore (schermo diviso)
5	Testo a 80 colonne

Figura 6-4. Modalità grafiche

Il parametro CLR significa cancellare. La Figura 6-5 spiega i valori associati a CLR.

Valore C	Descrizione
0	**Non cancella lo schermo grafico**
1	**Cancella lo schermo grafico**

Figura 6-5. Parametri CLR

Alla prima esecuzione del programma, se si desidera cancellare per la prima volta lo schermo grafico, impostare a 1 il parametro c nel comando GRAPHIC. Alla seconda esecuzione, per mantenere la figura sullo schermo, senza doverla ridisegnare ogni volta, impostare a 0 il parametro c.

Il parametro s specifica l'inizio dello schermo di testo in modalità a schermo diviso (lo schermo di testo inizierà al numero di riga successivo a quello specificato). Tralasciando il parametro s e selezionando una modalità grafica a schermo diviso (2 o 4) la porzione dello schermo di testo verrà visualizzata a partire dalla riga 20 fino alla riga 25; la parte rimanente dello schermo sarà a matrice di punti. Il parametro s permette di cambiare la riga iniziale dello schermo di testo in una riga qualsiasi dello schermo, da 1 a 25. Uno zero come parametro s indica che lo schermo non è diviso ed è interamente di testo.

L'ultimo parametro del comando GRAPHIC è CLR. Quando viene dato per la prima volta un comando grafico a matrice di punti, il Commodore 128 riserva un'area di 9K per l'informazione di schermo a matrice di punti. 8K vengono riservati ai dati della matrice di punti e 1K addizionale viene dedicato ai dati del colore (matrice video). Essendo 9K un blocco di memoria piuttosto esteso, lo si potrà utilizzare nuovamente più avanti per altri scopi in un programma. Questo è lo scopo di CLR, cioè di riorganizzare la memoria del Commodore 128 e di mettere nuovamente a disposizione per altri utilizzi i 9K di memoria che erano stati precedentemente usati per lo schermo a matrice di punti.

Il formato di CLR è il seguente:

GRAPHIC CLR

Quando si utilizza questo formato, occorre omettere tutti gli altri parametri del comando GRAPHIC.

Aggiungere il seguente comando al programma. Questo comando pone il C128 in modalità standard a matrice di punti e mette a disposizione uno schermo a matrice di punti a 8K (e 1K di dati del colore) per la creazione di grafici.

```
40 GRAPHIC 1,1
```

Il secondo 1 in questo comando cancella lo schermo a matrice di punti. Per non cancellare lo schermo, sostituire il secondo 1 con uno 0 (oppure toglierlo completamente).

> **NOTA:** Se si è impossibilitati a passare dalla modalità a matrice di punti allo schermo di testo, premere contemporaneamente i tasti RUN/STOP e RESTORE, oppure premere il tasto ESC seguito da X per tornare allo schermo a 80 colonne. Sebbene con il chip VIC (a 40 colonne) sia possibile solamente la **visualizzazione** di grafici, è tuttavia possibile la **scrittura** di programmi grafici nel formato a 80 colonne. Se si possiede il monitor duale Commodore 1901 e si desidera visualizzare il programma grafico quando è in fase di esecuzione, selezionare l'output a 40 colonne portando l'interruttore a cursore su output a 40 colonne.

Visualizzazione di grafici sullo schermo

Fino ad ora si è selezionata la modalità grafica ed i colori desiderati. Ora si può cominciare a visualizzare i grafici sullo schermo. Iniziare con il disegno di un cerchio.

Tracciamento di un cerchio - Il comando CIRCLE

Per disegnare un cerchio, utilizzare l'istruzione CIRCLE come mostrato di seguito.

```
60 CIRCLE 1,160,100,40,40
```

Questo visualizzerà un cerchio al centro dello schermo. L'istruzione CIRCLE possiede nove parametri che vengono selezionati per creare vari tipi di cerchi e di figure geometriche. Ad esempio, cambiando i numeri dell'istruzione CIRCLE nella riga 60 si otterranno dei cerchi di misure differenti oppure delle variazioni nella loro forma (es. un ovale). L'istruzione CIRCLE aggiunge potenza e versatilità alla programmazione di grafici in BASIC con il Commodore 128. Il significato dei numeri all'interno dell'istruzione CIRCLE viene spiegata nel paragrafo riguardante il comando CIRCLE nel Capitolo V, Enciclopedia BASIC 7.0.

Sullo schermo del Commodore 128, il punto in cui X = 0 e Y = 0 si trova nell'angolo superiore sinistro dello schermo e viene chiamato posizione HOME. Nella geometria convenzionale, comunque, il punto in cui X e Y sono uguali a 0 è l'angolo inferiore sinistro di un diagramma. La Figura 6-6 mostra l'introduzione delle coordinate di schermo X (orizzontale) e Y (verticale) ed i quattro punti agli angoli dello schermo del C128.

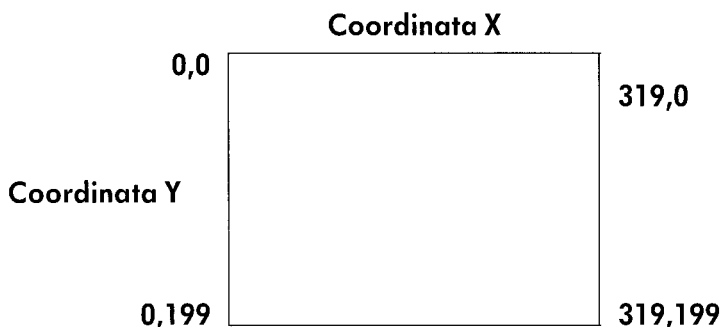



Figura 6-6. Disposizione delle coordinate X e Y.

Ecco il significato dei numeri:

- 1 è la sorgente di colore (in questo caso il primo piano)
- 160 è la coordinata iniziale X (orizzontale)
- 110 è la coordinata iniziale Y (verticale)
- 40 è il raggio

Tracciamento di un riquadro - Il comando BOX

Ora disegnare un riquadro battendo:

80 BOX 1,20,60,100,140,0,1

Questo disegna un riquadro a sinistra del cerchio. Per scoprire il significato dei numeri nell'istruzione per la creazione del riquadro, consultare il Capitolo V, Enciclopedia BASIC 7.0. L'istruzione BOX possiede sette parametri che vengono selezionati e modificati per produrre diversi tipi di riquadri. Cambiare il colore del primo piano e disegnare il contorno di un riquadro a destra del cerchio

usando le seguenti istruzioni:

```
90 COLOR1,9:REM MODIFICARE IL COLORE DEL PRIMO
   PIANO
100 BOX1,220,60,300,140,0,0
```

Utilizzando l'istruzione BOX produrre diverse combinazioni di rettangoli e riquadri.

Creazione di linee, punti ed altre figure - Il comando DRAW

Fino ad ora si è visto come selezionare modalità grafiche e colori e come visualizzare sullo schermo cerchi e riquadri. Un'altra istruzione grafica, DRAW, permette di disegnare linee sullo schermo esattamente come si farebbe con una matita su un foglio di carta. L'istruzione seguente permette di tracciare una linea al di sotto dei riquadri e del cerchio.

```
120 DRAW 1,20,180 TO 300,180
```

Per cancellare una linea, cambiare la sorgente (1) in 0 nell'istruzione DRAW. Sulla riga viene sovrapposto il colore dello sfondo provocando così la sua cancellazione. Provare ad utilizzare diverse coordinate ed altre sorgenti per acquisire dimestichezza con l'istruzione DRAW.

L'istruzione DRAW può essere usata in un altro formato con il quale è possibile tracciare una riga, modificare la direzione e tracciare un'altra riga in modo che le righe siano continue. Ad esempio, provare la seguente istruzione:

```
130 DRAW 1,250,0 TO 30,0 TO 40,40 TO 250,0
```

L'istruzione disegna un triangolo sulla parte superiore dello schermo. I quattro gruppi di numeri rappresentano le coordinate X e Y dei tre punti del triangolo. Notare che la prima e l'ultima coordinata sono identiche in quanto il disegno del triangolo deve terminare nello stesso punto in cui è iniziato. Questo tipo di istruzione DRAW permette di disegnare quasi tutte le forme geometriche come ad esempio trapezoidi, parallelogrammi e poligoni.

L'istruzione DRAW può essere usata in un terzo formato.

Ad esempio, per disegnare un punto, specificare i valori iniziali X e Y come segue:

```
150 DRAW 1,160,160
```

Questa istruzione disegna un punto al di sotto del cerchio.

Come si è visto, l'istruzione DRAW possiede caratteristiche versatili che permettono di creare sullo schermo figure, righe, punti e praticamente un numero illimitato di disegni.

Colorazione di aree delimitate - Il comando PAINT

L'istruzione DRAW permette di tracciare i contorni di aree sullo schermo. Per colorare le aree all'interno delle linee tracciate, viene utilizzata l'istruzione PAINT. Questa istruzione infatti permette di riempire di colore le aree delineate. Esattamente come un pittore dipinge sulla tela, allo stesso modo l'istruzione PAINT copre l'area dello schermo con i 16 colori disponibili. Ad esempio, battere:

160 PAINT 1,150,97

La riga 160 dipinge il cerchio che è stato tracciato nella riga 60. L'istruzione PAINT riempie un'area definita fino a quando non viene incontrato un limite specifico a seconda della sorgente indicata. Quando il Commodore 128 termina l'operazione di colorazione, lascia il cursore pixel sul punto in cui la colorazione era iniziata (cioè al punto 150,97).

Di seguito vengono mostrate altre istruzioni PAINT:

180 PAINT 1,50,25
200 PAINT 1,225,125

La riga 180 colora il triangolo e la riga 200 colora il riquadro.

> * **CONSIGLIO UTILE:** Se nell'istruzione PAINT si sceglie un punto iniziale che è già stato colorato dalla stessa sorgente, il Commodore 128 non dipingerà quell'area. Occorre scegliere un punto iniziale che si trova completamente all'interno del perimetro della figura che si desidera colorare. Il punto iniziale non può essere sulla riga del perimetro di un pixel che è stato colorato dalla stessa sorgente. I numeri di sorgente della coordinata di schermo e la coordinata specifica nel comando PAINT devono essere diversi tra loro.

Visualizzazione di caratteri su uno schermo a matrice di punti - Il comando CHAR

Fino ad ora l'esempio di programma è stato utilizzato in modalità standard a matrice di punti. Questa modalità usa un'area di memoria totalmente differente per la memorizzazione dei dati di schermo di quella per la memorizzazione della modalità testo (la

modalità in cui vengono introdotti programmi e testo). Attivando la modalità a matrice di punti e cercando di battere alcuni caratteri, non si ottiene alcun risultato. Questo perchè i caratteri che si stanno battendo possono essere visualizzati sullo schermo di testo e non sullo schermo a matrice di punti attivo in questo momento. A volte può essere necessario visualizzare caratteri su uno schermo a matrice di punti durante la creazione di diagrammi e grafici. Il comando CHAR è stato studiato appositamente per questo scopo. Per visualizzare i caratteri standard su uno schermo a matrice di punti, utilizzare l'istruzione CHAR come segue:

220 CHAR 1,11,24,"ESEMPIO GRAFICO"

Questo visualizza il testo "ESEMPIO GRAFICO" a partire dalla riga 25, colonna 12. Il comando CHAR può inoltre essere utilizzato in modalità testo, nonostante sia stato concepito per lo schermo a matrice di punti.

Creazione di un programma grafico

Fino a questo momento si è visto come utilizzare le diverse istruzioni grafiche. Ora, comporre il programma ed osservare come le varie istruzioni funzionano. Il programma sarà come mostrato di seguito. Le istruzioni per il colore nelle righe 70, 110, 140, 170, 190 e 210 sono state aggiunte per visualizzare ogni figura con un colore differente.

```
10 COLOR 0,1                         :REM SELECT BACKGROUND COLOR
20 COLOR 1,3                         :REM SELECT FOREGROUND COLOR
30 COLOR 4,1                         :REM SELECT BORDER COLOR
40 GRAPHIC 1,1                       :REM SELECT BIT MAP MODE
60 CIRCLE 1,160,100,40,40            :REM DRAW A CIRCLE
70 COLOR 1,6                         :REM CHANGE FOREGROUND COLOR
80 BOX 1,20,60,100,140,0,1           :REM DRAW A BLOCK
90 COLOR 1,9                         :REM CHANGE FOREGROUND COLOR
100 BOX 1,220,60,300,140,0,0         :REM DRAW A BOX
110 COLOR 1,8                        :REM CHANGE FOREGROUND COLOR
120 DRAW 1,20,180 TO 300,180         :REM DRAW A LINE
130 DRAW 1,250,0 TO 30,0 TO 40,40 TO 250,0:REM DRAW A TRIANGLE
140 COLOR 1,15                       :REM CHANGE FOREGROUND COLOR
150 DRAW 1,160,160                   :REM DRAW A POINT
160 PAINT 1,150,97                   :REM PAINT IN CIRCLE
170 COLOR 1,5                        :REM CHANGE FOREGROUND COLOR
180 PAINT 1,50,25                    :REM PAINT IN TRIANGLE
190 COLOR 1,7                        :REM CHANGE FOREGROUND COLOR
200 PAINT 1,225,125                  :REM PAINT IN EMPTY BOX
210 COLOR 1,11                       :REM CHANGE FOREGROUND COLOR
220 CHAR 1,11,24,"GRAPHICS EXAMPLE":REM DISPLAY TEXT
230 FOR I=1 TO 5000:NEXT:GRAPHIC 0,1:COLOR 1,2
```

Di seguito vengono elencate le varie fasi seguite dal programma:

- Le righe da 10 a 30 selezionano rispettivamente il colore dello sfondo, del primo piano e della cornice.
- La riga 40 sceglie una modalità grafica.
- La riga 60 visualizza un cerchio.
- La riga 80 disegna un riquadro colorato.
- La riga 100 traccia il perimetro di un riquadro.
- La riga 120 traccia una riga in fondo allo schermo.
- La riga 130 disegna un triangolo.
- La riga 150 produce un punto al di sotto del cerchio.
- La riga 160 colora il cerchio.
- La riga 180 colora il triangolo.
- La riga 200 colora il riquadro.
- La riga 220 stampa i caratteri "ESEMPIO GRAFICO" sul fondo dell schermo.
- La riga 230 ritarda il programma in modo che le figure rimangano visualizzate per un certo periodo di tempo sullo schermo, ritorna alla modalità testo e colora di nero i caratteri.

Se si desidera che le figure rimangano visualizzate sullo schermo, omettere l'istruzione GRAPHIC nella riga 230.

Modifica della dimensione delle immagini grafiche - Il comando SCALE

Il Commodore 128 possiede un'altra istruzione grafica che dà maggiore potenzialità al proprio sistema grafico. L'istruzione SCALE permette di aumentare o ridurre la dimensione delle immagini grafiche sullo schermo. L'istruzione SCALE ha un'altra funzione, che viene spiegata di seguito.

In modalità standard a matrice di punti, lo schermo a 40 colonne ha 320 coordinate orizzontali e 200 coordinate verticali. Nella modalità multicolore a matrice di punti, lo schermo a 40 colonne possiede un risoluzione che è solo la metà di quella della modalità standard a matrice di punti, cioè 160 x 200. Questa riduzione di risoluzione viene compensata dal fatto di poter utilizzare un colore addizionale per un totale di tre colori, all'interno di una matrice di carattere 8 per 8. La modalità standard a matrice di punti può visualizzare solo due colori all'interno di una matrice di carattere 8 per 8.

Quando si usa l'istruzione SCALE, le modalità standard a matrice di punti e multicolore a matrice di punti hanno le coordinate proporzionali l'una con l'altra. La scala va da 0 ad un massimo di 1023 coordinate orizzontali. Questo si verifica indipendentemente dal fatto che ci si trovi in modalità standard a matrice di punti o multicolore.

Per attivare la funzione SCALE, battere:

SCALE 1,x,y

Si otterranno così coordinate di schermo da 0 a 65535 valide sia in modalità standard sia ad alta risoluzione multicolore. Per disattivare SCALE, battere:

SCALE 0

Le coordinate ritorneranno ai valori normali.

Per vedere l'effetto di SCALE nel programma, aggiungere la riga 50:

50 SCALE 1,500,500

ed eseguirlo.

Per maggiori dettagli sul comando SCALE vedere il Capitolo V.
Nota: SCALE viene utilizzato dopo GRAPHIC e non ha alcun effetto su CHAR.

Di seguito vengono mostrati alcuni esempi di programma che utilizzano le istruzioni grafiche di cui sopra.

```
10 COLOR 0,1
20 COLOR 1,8
30 COLOR 4,1
40 GRAPHIC 1,1
50 FOR I=80 TO 240 STEP 10
60 CIRCLE 1,I,100,75,75
70 NEXT I
80 COLOR 1,5
90 FOR I= 80 TO 250 STEP 10
100 CIRCLE 1,I,100,50,50
110 NEXT I
120 COLOR 1,7
130 FOR I=50 TO 280 STEP 10
140 CIRCLE 1,I,100,25,25
150 NEXT I
160 FOR I=1 TO 7500:NEXT I
170 GRAPHIC 0,1:COLOR 1,2
```

```
10 GRAPHIC 1,1
20 COLOR 0,1
30 COLOR 4,1
40 FOR I=1 TO 50
50 Z=INT (((RND (1))*16)+1)*1
60 COLOR 1,Z
70 X=INT (((RND (1))*30)+1)*10
80 Y=INT (((RND (1))*20)+1)*10
90 U=INT (((RND (1))*30)+1)*10
100 V=INT (((RND (1))*20)+1)*10
110 DRAW 1,X,Y TO U,V
120 NEXT I
130 SCNCLR
140 GOTO 40
```

```
10 COLOR 4,7:COLOR 0,7:COLOR 1,1
20 GRAPHIC 1,1
30 FOR I=400 TO 1 STEP -5
40 DRAW 1,150,100 TO I,1
50 NEXT I
60 FOR I=1 TO 400 STEP 5
70 DRAW 1,150,100 TO 1,I
80 NEXT I
90 FOR I=40 TO 320 STEP 5
100 DRAW 1,150,100 TO I,320
110 NEXT I
120 FOR I=320 TO 30 STEP -5
130 DRAW 1,150,100 TO 320,I
140 NEXT I
150 FOR I=1 TO 7500:NEXT I
160 GRAPHIC 0,1:COLOR 1,1
```

Battere gli esempi, eseguirli e salvarli per un uso futuro. Uno dei migliori modi per l'apprendimento della programmazione è lo studio degli esempi di programma per capire come le diverse istruzioni eseguono le proprie funzioni. In questo modo si potranno utilizzare con facilità le istruzioni grafiche per creare proprie figure con il Commodore 128.

Per maggiori dettagli sulle istruzioni o sui comandi BASIC, consultrare l'Enciclopedia BASIC 7.0 nel Capitolo V.

A questo punto si ha così a disposizione una serie di comandi grafici che permettono di creare un numero quasi illimitato di visualizzazioni grafiche. Le caratteristiche grafiche del Commodore 128 però non finiscono qui. Infatti il Commodore 128 possiede un'altra serie di istruzioni, chiamate grafici SPRITE, che rendono veloci, semplici e sofisticati la creazione ed il controllo delle immagini grafiche. Queste istruzioni ad alto livello permettono di creare sprite (figure mobili). Il C128 possiede la propria funzione incorporata di definizione sprite (SPRDEF). Queste istruzioni rappresentano la nuova tecnologia per la creazione ed il movimento degli sprite. Leggere la sezione seguente nella quale verranno date le prime nozioni sull'animazione con il computer.

Sprite: figure animate programmabili

Si è visto fino ad ora come utilizzare le funzioni grafiche del Commodore 128, come usare la prima serie di istruzioni grafiche ad alto livello per tracciare cerchi, riquadri, righe e punti, ed inoltre come colorare lo schermo, attivare modalità grafiche, colorare oggetti sullo schermo e modificarne le dimensioni in modo proporzionale. Ora si vedrà la prossima fase di programmazione grafica: l'animazione degli sprite.

Coloro che hanno già usato il Commodore 64 sicuramente avranno già sentito parlare degli sprite; per coloro che invece non hanno dimestichezza con questo termine, uno sprite è una figura animata creata dall'utente. Gli sprite possono essere colorati con 16 colori diversi oppure possono essere multicolori. Queste figure possono essere mosse sullo schermo.

Di seguito viene data una lista delle istruzioni che verranno spiegate in questa sezione:

MOVRSPR
SPRDEF
SPRITE
SPRSAV
SSHAPE

Creazione di sprite

Per prima cosa si dovrà disegnare la forma dello sprite. Ad esempio, supponiamo di voler disegnare una nave spaziale oppure una macchina da corsa. Prima di colorare o muovere lo sprite si dovrà definire la sua immagine. In modalità C128 gli sprite possono essere creati in tre modi differenti:

1. Utilizzando l'istruzione SPRITE all'interno di un programma.
2. Utilizzando la modalità di definizione sprite (SPRDEF).
3. Utilizzando lo stesso metodo del Commodore 64.

Uso delle istruzioni sprite in un programma

Questo metodo, al contrario degli altri due, usa le istruzioni incorporate e alcune delle istruzioni grafiche apprese nella sezione precedente, come mostrato nella procedura generale di seguito. I particolari verranno aggiunti mano a mano che si prosegue nella sezione.

1. Disegnare un'immagine per mezzo delle istruzioni grafiche apprese nell'ultima sezione, come ad esempio DRAW, CIRCLE, BOX e PAINT. Le dimensioni dovranno essere 24 pixel di larghezza per 21 pixel di altezza se in modalità standard a matrice di punti oppure 12 pixel di larghezza per 21 pixel di altezza se in modalità multicolore a matrice di punti.
2. Usare l'istruzione SSHAPE per memorizzare i dati della figura in una variabile stringa.
3. Trasferire i dati della figura dalla variabile stringa allo sprite per mezzo dell'istruzione SPRSAV.
4. Attivare lo sprite, colorarlo, selezionare la modalità standard o multicolor ed espanderlo, tutto questo con l'istruzione SPRITE.
5. Muovere lo sprite con l'istruzione MOVRSPR.

Disegno dello sprite

Di seguito vengono illustrate le istruzioni che eseguono le operazioni concernenti gli sprite. Alla fine di questa sezione, come risultato si avrà il primo programma di sprite che potrà essere eseguito e salvato per uso futuro.

Per prima cosa si disegnerà sullo schermo una figura (24 pixel per 21) per mezzo di DRAW, CIRCLE, BOX o PAINT. Questo esempio viene eseguito in modalità standard a matrice di punti, con lo sfondo di colore nero. L'istruzione di impostazione della modalità grafica e di colorazione in nero dello sfondo è la seguente:

```
5 COLOR 0,1:REM COLORA LO SFONDO DI NERO
10 GRAPHIC 1,1:REM IMPOSTA MODALITÀ STANDARD A
   MATRICE DI PUNTI
```

Le istruzioni seguenti disegnano una figura di un'auto da corsa nell'angolo superiore sinistro dello schermo. Queste istruzioni sono già state spiegate nella sezione precedente.

```
5 COLOR 0,1:COLOR 4,1:COLOR 1,2  :REM SET COLORS
10 GRAPHIC 1,1                   :REM SET HI-RES GRAPHIC MODE
15 BOX 1,2,2,45,45               :REM PICTURE FRAME
20 DRAW 1,17,10 TO 28,10 TO 26,30:REM CAR BODY
22 DRAWTO 19,30 TO 17,10         :REM CAR BODY
24 BOX 1,11,10,15,18             :REM UPPER LEFT WHEEL
26 BOX 1,30,10,34,18             :REM UPPER RIGHT WHEEL
28 BOX 1,11,20,15,28             :REM LOWER LEFT WHEEL
30 BOX 1,30,20,34,28             :REM LOWER RIGHT WHEEL
32 DRAW 1,26,28 TO 19,28         :REM DRAW GRILLE
34 BOX 1,20,14,26,18,90,1        :REM CAR SEAT
36 BOX 1,150,35,195,40,90,1      :REM WHITE LINES
38 BOX 1,150,135,195,140,90,1    :REM WHITE LINES
40 BOX 1,150,215,195,220,90,1    :REM WHITE LINES
42 BOX 1,50,180,300,195          :REM FINISH OUTLINE
44 CHAR 1,18,23,"FINISH"         :REM DISPLAY FINISH
```

Eseguire il programma. In questo modo si è disegnata un'auto da corsa bianca, racchiusa in un riquadro, nell'angolo superiore sinistro dello schermo. Inoltre si è disegnata una pista con il traguardo in fondo allo schermo. A questo punto, l'auto è solo una figura ferma sullo schermo e non è ancora uno sprite. La prima fase di programmazione dello sprite è completata: l'immagine è stata definita.

Memorizzazione dei dati dello sprite con SSHAPE

La prossima fase sarà il salvataggio della figura in una stringa di testo per mezzo dell'istruzione seguente:

45 SSHAPE A$,11,10,34,30:REM SALVA LA FIGURA IN UNA STRINGA

Il comando SSHAPE memorizza l'immagine dello schermo (configurazione di bit) in una variabile stringa che sarà utilizzata in futuro secondo le coordinate di schermo specificate.

I numeri 11,10,34,30 sono le coordinate della figura. Se queste coordinate non vengono introdotte nel punto esatto, l'istruzione SSHAPE non potrà memorizzare correttamente i dati dell'immagine nella variabile stringa A$. Se l'istruzione SSHAPE viene posizionata su una locazione di schermo vuota, la stringa dati sarà vuota, come si noterà al trasferimento della stringa in uno sprite. Assicurarsi di aver introdotto l'istruzione SSHAPE direttamente nella coordinata esatta. Inoltre, verificare le dimensioni corrette del singolo sprite, che dovranno essere 24 pixel di larghezza per 21 pixel di altezza.

L'istruzione SSHAPE trasferisce la figura dell'auto da corsa in una stringa dati che il computer interpreterà come dati della figura stessa. La stringa dati, A$, memorizza nella memoria del computer una stringa di zeri e di uno che comporrà la figura sullo schermo. Come per tutte le funzioni grafiche di tutti i computer, il computer stesso ha un modo per rappresentare visivamente la grafica con i bit in memoria. Ogni puntino dello schermo, chiamato pixel, ha un bit nella memoria del computer che lo controlla. Nella modalità standard a matrice di punti, se il bit nella memoria è uguale a 1 (attivo), il pixel sullo schermo sarà attivato. Se il bit di controllo nella memoria è uguale a 0 (disattivato) il pixel sarà disattivato.

Memorizzazione dei dati della figura in uno sprite

La figura è ora memorizzata in una stringa. La prossima fase sarà trasferire i dati della figura dalla stringa dati (A$) all'area dati dello sprite in modo da attivare lo sprite e animarlo. L'istruzione per questa operazione è SPRSAV.

```
50 SPRSAV A$,1:REM MEMORIZZA LA STRINGA DATI
   NELLO SPRITE 1
55 SPRSAV A$,2:REM MEMORIZZA LA STRINGA DATI
   NELLO SPRITE 2
```

I dati della figura saranno trasferiti negli sprite 1 e 2. Entrambi questi sprite hanno gli stessi dati, quindi hanno esattamente lo stesso aspetto. Gli sprite non possono ancora essere visualizzati in quanto devono essere ancora attivati.

Attivazione degli sprite

L'istruzione SPRITE attiva uno sprite specifico (numerato da 1 a 8), lo colora, specifica la sua priorità di schermo, ne espande le dimensioni e determina il tipo di visualizzazione dello sprite. La priorità di schermo determina se lo sprite passerà davanti o dietro agli oggetti sullo schermo. Gli sprite possono essere espansi fino al doppio della loro dimensione originale sia in larghezza sia in altezza. Il tipo di visualizzazione di sprite determina se lo sprite è uno sprite standard a matrice di punti o uno sprite multicolor a matrice di punti. Di seguito vengono mostrate due istruzioni che attivano gli sprite 1 e 2.

```
60 SPRITE 1,1,7,0,0,0,0:REM ATTIVA SPR 1
65 SPRITE 2,1,3,0,0,0,0:REM ATTIVA SPR 2
```

Il significato dei numeri nelle istruzioni SPRITE è il seguente:

```
SPRITE #,A,C,P,X,Y,M
```

\# - Numero dello sprite (da 1 a 8)
A - Attivo (A = 1) o disattivo (A = 0)
C - Colore (da 1 a 16)
P - Priorità - Se P = 0, lo sprite passa davanti agli oggetti sullo schermo
Se P = 1, lo sprite passa dietro agli oggetti sullo schermo
X - Se X = 1, lo sprite viene espanso in larghezza (X)
Se X = 0, lo sprite sarà in larghezza (X)
Y - Se Y = 1, lo sprite viene espanso in altezza (Y)
Se Y = 0, lo sprite sarà in altezza normale
M - Se M = 1, lo sprite sarà multicolore
Se M = 0, lo sprite sarà standard

Come si sarà notato, l'istruzione SPRITE è molto potente in quanto permette di avere il controllo su molte delle caratteristiche degli sprite.

Movimento degli sprite con MOVSPR

A questo punto verrà mostrato come muovere lo sprite sullo schermo. L'istruzione MOVSPR controlla il movimento di uno sprite e permette di animarlo sullo schermo. L'istruzione MOVSPR può essere usata in due modi. Per prima cosa l'istruzione MOVSPR può posizionare uno sprite in un punto preciso dello schermo per mezzo delle coordinate orizzontali e verticali. Aggiungere le seguenti istruzioni al programma:

```
70 MOVSPR 1,240,70:REM POSIZIONA LO SPRITE
   1-X=240, Y=70
80 MOVSPR 2,120,70:REM POSIZIONA LO SPRITE
   2-X=120, Y=70
```

La riga 70 posiziona lo sprite 1 alla coordinata 240,70. La riga 80 posiziona lo sprite 2 alla coordinata 120,70. L'istruzione MOVSPR può anche essere utilizzata per muovere gli sprite a seconda della loro posizione originale sullo schermo. Ad esempio, posizionare gli sprite 1 e 2 alle coordinate indicate nelle righe 70 e 80. Per spostarli dalle posizioni originali ad un altro punto sullo schermo, usare le seguenti istruzioni:

```
85 MOVSPR 1,180 # 6:REM MUOVE LO SPRITE 1 DALLA
   PARTE SUPERIORE DELLO SCHERMO ALLA PARTE
   INFERIORE
87 MOVSPR 2,180 # 7:REM MUOVE LO SPRITE 2 DALLA
   PARTE SUPERIORE DELLO SCHERMO ALLA PARTE
   INFERIORE
```

Il primo numero in queste istruzioni è il numero di sprite. Il secondo numero è la direzione in senso orario espressa in gradi in relazione alla posizione originale dello sprite. Il simbolo # significa che lo sprite viene mosso ad un angolo specifico e ad una velocità relativa alla posizione iniziale, invece che ad una posizione assoluta, come indicato nelle righe 70 e 80. L'ultimo numero specifica la velocità con la quale lo sprite si muove lungo la traettoria sullo schermo. Questa velocità è espressa con un valore da 0 a 15.

Il comando MOVSPR ha due formati alternativi. Per dettagli, consultare l'Enciclopedia BASIC 7.0 nel Capitolo V.

Gli sprite utilizzano delle coordinate completamente diverse da quelle della matrice di punti. Le coordinate della matrice di punti sono comprese in una gamma da 0,0 punti (l'angolo superiore sinistro) a 319,199 (l'angolo inferiore destro). Le coordinate visibili dello sprite iniziano al punto 50,24 e terminano al punto 250,344. La parte rimanente delle coordinate di sprite sono al di fuori dello schermo e non sono visibili, però lo sprite si muove ugualmente su queste coordinate. Le locazioni fuori campo permettono agli sprite di muoversi facilmente dentro e fuori lo schermo. La Figura 6-7 mostra le coordinate di sprite e le posizioni visibili di sprite.

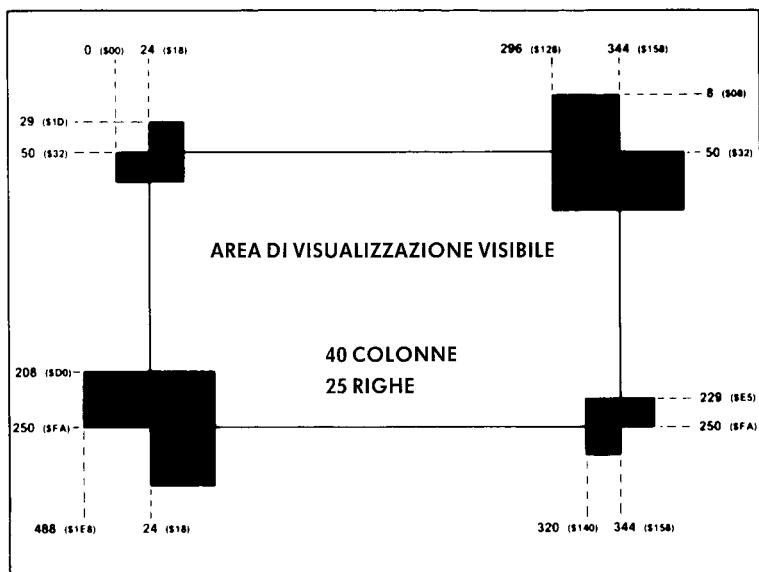



Figura 6-7. Coordinate visibili di sprite

Ora eseguire l'intero programma. In questo modo è stato scritto il primo programma: una gara automobilistica con due auto. Provare ad aggiungere più auto e più figure sullo schermo. Disegnare altri sprite ed includerli nella gara. Immaginare altre scene ed altre figure da animare per poter creare molti "film" animati con il computer.

Per arrestare gli sprite, premere contemporaneamente RUN/STOP e RESTORE.

Creazione di un programma di sprite

Si è così creato un esempio di programma di sprite il cui listato è il seguente:

```
5 COLOR 0,1:COLOR 4,1:COLOR 1,2   :REM SET COLORS
10 GRAPHIC 1,1                     :REM SET HI-RES GRAPHIC MODE
15 BOX 1,2,2,45,45                 :REM PICTURE FRAME
20 DRAW 1,17,10 TO 28,10 TO 26,30:REM CAR BODY
22 DRAWTO 19,30 TO 17,10           :REM CAR BODY
24 BOX 1,11,10,15,18               :REM UPPER LEFT WHEEL
26 BOX 1,30,10,34,18               :REM UPPER RIGHT WHEEL
28 BOX 1,11,20,15,28               :REM LOWER LEFT WHEEL
30 BOX 1,30,20,34,28               :REM LOWER RIGHT WHEEL
32 DRAW 1,26,28 TO 19,28           :REM DRAW GRILLE
34 BOX 1,20,14,26,18,90,1          :REM CAR SEAT
36 BOX 1,150,35,195,40,90,1        :REM WHITE LINES
38 BOX 1,150,135,195,140,90,1      :REM WHITE LINES
40 BOX 1,150,215,195,220,90,1      :REM WHITE LINES
42 BOX 1,50,180,300,195            :REM FINISH OUTLINE
44 CHAR 1,18,23,"FINISH"           :REM DISPLAY FINISH
45 SSHAPE A$,11,10,34,30           :REM SAVE PICTURE INTO A$
50 SPRSAV A$,1                     :REM STORE A$ IN SPRITE 1
55 SPRSAV A$,2                     :REM STORE A$ IN SPRITE 2
60 SPRITE 1,1,7,0,0,0,0            :REM TURN ON SPRITE 1
65 SPRITE 2,1,3,0,0,0,0            :REM TURN ON SPRITE 2
70 MOVSPR 1,240,70                 :REM SPRITE 1 X=240, Y=70
80 MOVSPR 2,120,70                 :REM SPRITE 2 X=120, Y=70
85 MOVSPR 1,180 #6                 :REM MOVE SPR 1 DOWN SCREEN
87 MOVSPR 2,180 #7                 :REM MOVE SPR 2 DOWN SCREEN
90 FOR I=1 TO 5000:NEXT I
99 GRAPHIC 0,1
```

Di seguito vengono elencate le varie fasi seguite dal programma:

- La riga 5 colora lo schermo di nero.
- La riga 10 imposta la modalità grafica standard ad alta risoluzione.

- La riga 15 disegna un riquadro nell'angolo superiore sinistro dello schermo.
- Le righe da 20 a 32 disegnano l'auto da corsa.
- Le righe da 35 a 44 disegnano le corsie ed il traguardo.
- La riga 45 trasferisce i dati della figura dall'auto alla variabile stringa.
- Le righe 50 e 55 trasferiscono il contenuto della variabile stringa agli sprite 1 e 2.
- Le righe 60 e 65 attivano gli sprite 1 e 2.
- Le righe 70 e 80 posizionano gli sprite sulla parte superiore dello schermo.
- Le righe 85 e 87 animano gli sprite come se le due auto stessero passando il traguardo.

In questa sezione si è appreso come creare sprite per mezzo delle istruzioni grafiche incorporate del C128, come ad esempio DRAW e BOX e come controllare gli sprite usando le istruzioni del Commodore 128. Il Commodore 128 mette a disposizione due altri modi per poter creare gli sprite: il primo è la funzione incorporata di definizione sprite (SPRDEF), come descritto nei paragrafi seguenti; il secondo è lo stesso utilizzato dal Commodore 64. Consultare la Guida di Riferimento per il Programmatore del C64 per dettagli su quest'ultima tecnica di creazione di sprite.

Modalità di definizione sprite - Il comando SPRDEF

Il Commodore 128 possiede una modalità incorporata di definizione sprite che permette di creare sprite con il Commodore 128. Con il Commodore 64 ad esempio per questa operazione occorre avere uno sprite editor addizionale oppure disegnare uno sprite su un foglio di carta millimetrata e quindi introdurre nel computer con i comandi READ, DATA e POKE lo sprite codificato. Con il nuovo comando SPRDEF di definizione sprite del Commodore 128 è possibile creare e modificare gli sprite in un'area di lavoro specifica.

Per attivare la modalità SPRDEF, battere:

SPRDEF

e premere RETURN. Il Commodore 128 visualizza sullo schermo un reticolo di sprite. Inoltre, il computer visualizza il messaggio:

SPRITE NUMBER?

Battere un numero da 1 a 8. Il computer riempie il reticolo e visualizza lo sprite corrispondente nell'angolo superiore destro dello schermo. D'ora in poi, il reticolo di sprite verrà chiamato area di lavoro. L'area di lavoro è larga 24 caratteri e alta 21 caratteri. Ogni posizione di carattere all'interno dell'area di lavoro corrisponde ad 1 pixel dello sprite, in quanto uno sprite è composto da 24 pixel di larghezza per 21 pixel di altezza. Nell'area di lavoro, in modalità SPRDEF sono disponibili diversi comandi di modifica. Di seguito viene dato un riassunto dei comandi:

Riassunto dei comandi in modalità definizione Sprite

Tasto CLR - cancella l'intera area di lavoro
Tasto M - Attiva/disattiva lo sprite multicolore
CTRL 1-8 - Seleziona il colore di primo piano dello sprite (da 1 a 8)
C= 1-8 - Seleziona il colore di primo piano dello sprite (da 9 a 16)
Tasto 1 - Attiva i pixel nel colore di sfondo
Tasto 2 - Attiva i pixel nel colore di primo piano
Tasto 3 - Attiva le aree in multicolor 1
Tasto 4 - Attiva le aree in multicolor 2
Tasto A - Attiva/disattiva il movimento automatico del cursore
Tasti CRSR - Muovono il cursore (+) all'interno dell'area di lavoro
RETURN - Muove il cursore all'inizio della riga sottostante
Tasto HOME - Muove il cursore all'angolo superiore sinistro dell'area di lavoro
Tasto X - Controlla l'espansione orizzontale
Tasto Y - Controlla l'espanzione verticale
Shift RETURN - Memorizza lo sprite dall'area di lavoro e visualizza nuovamente il prompt SPRITE NUMBER
Tasto C - Copia uno sprite in un altro
Tasto STOP - Disattiva lo sprite visualizzato e visualizza nuovamente il prompt SPRITE NUMBER senza modificare lo sprite
Tasto RETURN - (alla visualizzazione del prompt SPRITE NUMBER). Esce dalla modalità SPRDEF.

Procedura di creazione sprite in modalità definizione sprite (SPRDEF)

La procedura generale per la creazione di sprite in modalità definizione sprite è la seguente:

1. Cancellare l'area di lavoro premendo contemporaneamente i tasti shift e CLR/HOME.
2. Se si desidera uno sprite multicolor, premere il tasto M. Vicino al cursore normale apparirà un cursore addizionale. La ragione dei due cursori è data dal fatto che la modalità multicolor attiva due pixel per ogni singolo pixel in modalità standard. Ecco perchè la modalità multicolor è solo metà della risoluzione orizzontale della modalità ad alta risoluzione.
3. Selezionare il colore desiderato per lo sprite. Per i colori da 1 a 8 tenere premuto il tasto CONTROL contemporaneamente ad un tasto tra 1 e 8. Per selezionare i codici colore da 9 a 16 tenere premuto il tasto C= e premere un tasto da 1 a 8.
4. Si è ora pronti ad iniziare la creazione della forma dello sprite. I tasti numerati da 1 a 4 riempiono lo sprite e gli danno forma. Per uno sprite di un solo colore, usare il tasto 2 per riempire una posizione di carattere all'interno dell'area di lavoro. Premere il tasto 1 per cancellare quello che era stato tracciato con il tasto 2. Se si desidera riempire una posizione di carattere per volta, premere il tasto A. Ora si dovrà muovere manualmente il cursore per mezzo dei tasti di movimento cursore. Se si vuole spostare il cursore automaticamente verso destra mentre si preme un tasto controllo cursore, non premere il tasto A in quanto questo tasto è già impostato per il movimento automatico del cursore. Riempiendo una posizione di carattere all'interno dell'area di lavoro, si potrà notare che il pixel corrispondente nello sprite visualizzato si attiva. Gli sprite vengono modificati non appena si modifica l'area di lavoro.

   In modalità multicolore, il tasto 3 riempie due posizioni di caratteri nell'area di lavoro con il colore multicolor 1 ed il tasto 4 riempie due posizioni di caratteri con il multicolor 2.

   È possibile disattivare (colorare il pixel con il colore dello sfondo) le aree già riempite all'interno dell'area di lavoro premendo il tasto 1. In modalità multicolore, il tasto 1 disattiva due posizioni di caratteri con una sola pressione.

5. Durante la fase di costruzione dello sprite, è possibile muoversi liberamente all'interno dell'area di lavoro senza attivare o disattivare i pixel utilizzando i tasti RETURN, HOME ed i tasti di movimento cursore.

6. Gli sprite possono essere espansi in qualsiasi momento sia orizzontalmente sia verticalmente. Per espanderli verticalmente, premere il tasto Y. Per espanderli orizzontalmente, premere il tasto X. Per tornare alla dimensione normale di visualizzazione, premere nuovamente X o Y.

   Quando uno stesso tasto viene utilizzato per attivare E disattivare un funzione viene chiamato bistabile. I tasti X e Y operano in questo modo sull'espansione orizzontale e verticale dello sprite.

7. Quando si è finito di creare lo sprite e si è definita completamente la sua forma, occorre salvarlo tenendo premuto il tasto SHIFT e premendo il tasto RETURN. Il Commodore 128 salva i dati dello sprite nell'area di memorizzazione appropriata. Lo sprite visualizzato nell'angolo superiore destro dello schermo viene disattivato e quindi viene visualizzato il prompt SPRITE NUMBER. Per creare un altro sprite, introdurre un numero di sprite e modificare il nuovo sprite esattamente come si è fatto con il primo. Per visualizzare nuovamente lo sprite originale nell'area di lavoro, battere il numero dello sprite originale. Per uscire dalla modalità di definizione sprite, premere RETURN in risposta al prompt SPRITE NUMBER.

8. Per copiare uno sprite in un altro, utilizzare il tasto "C".

9. Se non si desidera salvare lo sprite, premere il tasto STOP. Il Commodore 128 disattiva lo sprite visualizzato e visualizza il messaggio SPRITE NUMBER.

10. Per uscire dalla modalità di definizione sprite, premere il tasto RETURN mentre sullo schermo è visualizzato il prompt SPRITE NUMBER senza alcun numero di sprite dopo di esso. È possibile uscire quando ci si trova in una delle due seguenti situazioni:

    Subito dopo aver salvato lo sprite (shift RETURN),
    Subito dopo aver premuto il tasto STOP

Dopo aver creato uno sprite ed essere usciti dalla modalità di definizione sprite, i dati dello sprite saranno memorizzati nell'area di

memorizzazione appropriata del Commodore 128. Per poter visualizzare lo sprite occorrerà attivarlo in quanto ci si trova in ambiente BASIC. Per attivare nuovamente lo sprite, usare il comando SPRITE come mostrato precedentemente. Ad esempio, se si era creato lo sprite 1 in modalità SPRDEF. Per attivare questo sprite in BASIC, per colorarlo di blu ed espanderlo sia orizzontalmente sia verticalmente, battere il comando seguente:

SPRITE 1,1,7,0,1,1,0

Ora utilizzare il comando MOVSPR per muovere lo sprite come mostrato di seguito:

MOVSPR, 1 45# 5

Questo è tutto per quanto riguarda la modalità SPRDEF. Per prima cosa occorre creare lo sprite, salvare i dati dello sprite ed uscire dalla modalità SPRDEV attivando il BASIC, quindi per attivare lo sprite utilizzare il comando SPRITE e per muoverlo usare il comando MOVSPR. Alla fine della fase di programmazione, salvare i dati dello sprite in un file binario per mezzo del comando BSAVE, come segue:

BSAVE "nomefile",B0, P3584 TO P4096

Per richiamare nuovamente i dati dello sprite dal disco, caricare il file binario salvato con BSAVE utilizzando il comando BLOAD come segue:

BLOAD "nomefile"[, B0, P3584]

La parte racchiusa tra parentesi quadre è opzionale. BLOAD carica i dati nell'indirizzo da cui erano stati salvati.

Si sono così appresi i metodi di creazione degli sprite: 1) SSHAPE, SPRSAV, SPRITE, MOVSPR, 2) MODALITÀ SPRDEF. Esercitarsi su questi metodi per perfezionarsi nelle animazioni degli sprite se non viene specificata la parte opzionale.

Per ulteriori informazioni consultare **Memorizzazione di dati di sprite in file binari** più avanti in questa sezione.

Unione di sprite

Fino ad ora si è appreso come creare, colorare, attivare ed animare gli sprite. Può capitare di voler creare una figura che però è troppo particolareggiata o troppo grande per poter essere racchiusa in un solo sprite. In questo caso è possibile collegare due o più sprite in modo che la figura sia più grande e più dettagliata di

quanto lo sarebbe utilizzando un singolo sprite. Per mezzo di questa funzione, gli sprite possono muoversi indipendentemente gli uni dagli altri, dando così la possibilità di avere più controllo sull'animazione.

In questa sezione viene fornito un esempio dell'uso di due sprite collegati. La procedura generale (algoritmo) per la scrittura di un programma con due o più sprite collegati è la seguente:

1. Disegnare una figura sullo schermo utilizzando le istruzioni grafiche del Commodore 128, quali DRAW, BOX e PAINT, esattamente come è stato fatto nel programma di gara automobilistica della sezione precedente. Questa volta però creare la figura con dimensioni doppie di quelle di uno sprite singolo, 48 pixel di larghezza per 21 pixel di altezza.
2. Utilizzare due istruzioni SSHAPE per memorizzare gli sprite in due stringhe di dati separate. Posizionare le coordinate della prima istruzione SSHAPE sull'area di 24 pixel per 21 pixel della prima metà della figura tracciata. Quindi posizionare le coordinate della seconda istruzione SSHAPE sulla seconda area di 24 pixel per 21 pixel. Assicurarsi di aver memorizzato i dati delle due metà della figura in due stringhe diverse. Ad esempio, la prima istruzione SSHAPE memorizza la prima metà della figura in A$, mentre la seconda istruzione SSHAPE memorizza la seconda metà della figura in B$.
3. Trasferire i dati della figura da ogni stringa di dati ad uno sprite separato per mezzo dell'istruzione SPRSAV.
4. Attivare ogni sprite con l'istruzione SPRITE.
5. Posizionare gli sprite in modo tale che il primo sprite inizi al pixel adiacente a quello in cui termina il secondo sprite. Con questa operazione gli sprite vengono collegati. Ad esempio, disegnare una figura di 48 pixel per 21 pixel. Posizionare il primo sprite (definito 1, ad esempio) sulle coordinate 10,10 come nell'istruzione seguente:

   100 MOVSPR 1,10,10

   dove il primo numero è il numero dello sprite, il secondo numero è la coordinata orizzontale (X) e il terzo numero è la coordinata verticale (Y). Posizionare il secondo sprite a destra di 24 pixel rispetto allo sprite 1 come mostrato:

   200 MOVSPR 2,34,10

A questo punto i due sprite vengono visualizzati uno accanto all'altro ed hanno lo stesso aspetto della figura disegnata all'inizio del programma con le istruzioni DRAW, BOX e PAINT.

6. Ora, per mezzo dell'istruzione MOVSPR è possibile muovere gli sprite insieme o in due direzioni differenti. Come si è appreso nella sezione precedente, l'istruzione MOVSPR permette di muovere gli sprite verso un punto specifico dello schermo oppure in una posizione relativa alla posizione originale dello sprite.

Il programma che segue è un esempio del collegamento di due sprite. Il programma crea una scena extraterrestre. Disegna stelle, un pianeta e una navicella spaziale simile a Apollo. La navicella viene disegnata e quindi memorizzata in due stringhe di dati, A$ e B$. La parte anteriore della navicella, la capsula, viene memorizzata nello sprite 1, mentre la parte posteriore della navicella, il missile, viene memorizzata nello sprite 2. La navicella vola lentamente attraversando lo schermo due volte. Essendo la sua velocità così ridotta, ed essendo la navicella molto lontana dalla Terra, si dovranno azionare i razzi posteriori per dirigerla verso la Terra. Dopo il secondo attraversamento dello schermo, verranno attivati i missili posteriori che spingeranno la navicella verso la Terra.

Il listato del programma sarà:

```
5 COLOR 4,1:COLOR 0,1:COLOR 1,2 :REM SELECT COLORS
10 GRAPHIC 1,1                   :REM SET HI-RES MODE
17 FOR I=1 TO 40
18 X=INT (RND (1)*320)+1
19 Y=INT (RND (1)*200)+1
21 DRAW 1,X,Y:NEXT I             :REM DRAW STARS
22 BOX 0,0,5,70,40,,1            :REM CLEAR BOX
23 BOX 1,1,5,70,40:COLOR 1,8     :REM BOX-IN SPACESHIP
24 CIRCLE1,190,90,35,25:PAINT1,190,95:REM DRAW AND PAINT PLANET
25 FOR I=90 TO 96 STEP 3:CIRCLE 1,190,I,65,10:NEXT I
26 DRAW 1,10,17TO16,17TO32,10TO33,20TO32,30TO16,23TO10,23TO10,17
28 DRAW 1,19,24TO20,21TO27,25TO26,28:REM BOTTOM WINDOW
35 DRAW 1,20,19TO20,17TO29,13TO30,18TO28,23TO20,19:REM TOP WINDOW
38 PAINT 1,13,20                 :REM PAINT SPACESHIP
40 DRAW 1,34,10TO36,20TO34,30TO45,30TO46,20TO45,10TO34,10
42 DRAW 1,45,10TO51,12TO57,10TO57,17TO51,15TO46,17:REM ENGINE 1
43 DRAW 1,46,22TO51,24TO57,22TO57,29TO51,27TO45,29:REM ENGINE 2
44 PAINT1,40,15:PAINT1,47,12:PAINT1,47,26:DRAW0,45,30TO46,20TO45,10
45 DRAW 0,34,14TO44,14:DRAW 0,34,21TO44,21:DRAW 0,34,28TO44,28
47 SSHAPE A$,10,10,33,32         :REM SAVE SPRITE IN A$
48 SSHAPE B$,34,10,57,32         :REM SAVE SPRITE IN B$
50 SPRSAV A$,1                   :REM SPRITE1 DATA
55 SPRSAV B$,2                   :REM SPRITE2 DATA
60 SPRITE 1,1,3,0,0,0,0          :REM ENABLE SPRITE 1 IN RED
65 SPRITE 2,1,7,0,0,0,0          :REM ENABLE SPRITE 2 IN BLUE
82 MOVSPR 1,150,150              :REM POSITION SPRITE1
83 MOVSPR 2,172,150              :REM POSITION SPRITE2
85 MOVSPR 1,270 #5               :REM ANIMATE SPRITE1
87 MOVSPR 2,270 #5               :REM ANIMATE SPRITE2
90 FOR I=1 TO 5000:NEXT I
92 MOVSPR 1,150,150              :REM RETRO POSITION
93 MOVSPR 2,174,150
95 MOVSPR 1,270 #10              :REM SPLIT SPRITES 1 & 2
96 MOVSPR 2,90 #5
97 FOR I=1 TO 1200:NEXT I
98 SPRITE 2,0                    :REM TURN OFF SPRITE 2
99 FORI=1 TO 5000:NEXT I
100 GRAPHIC 0,1                  :REM RETURN TO TEXT
```

Di seguito vengono elencate le varie fasi del programma:

- La riga 5 colora di nero lo sfondo e di bianco il primo piano.
- La riga 10 seleziona la modalità standard ad alta risoluzione e cancella lo schermo ad alta risoluzione.
- La riga 23 racchiude in un'area di visualizzazione la figura della navicella nell'angolo superiore sinistro dello schermo.
- Le righe da 17 a 21 disegnano le stelle.
- La riga 24 disegna e colora il pianeta.
- La riga 25 disegna gli anelli intorno al pianeta.
- La riga 26 disegna il contorno della capsula della navicella.

- La riga 28 disegna l'oblò inferiore della capsula spaziale.
- La riga 35 disegna l'oblò superiore della capsula spaziale.
- La riga 38 colora di bianco la capsula spaziale.
- La riga 40 disegna i contorni dei razzi posteriori della navicella.
- Le righe 42 e 43 disegnano i motori dei razzi posteriori sul retro della navicella.
- La riga 44 colora i motori dei razzi posteriori e disegna il contorno del fondo dei razzi posteriori nel colore dello sfondo.
- La riga 45 traccia delle righe sui razzi posteriori con il colore dello sfondo (fino a questo punto si sono visualizzate solo figure sullo schermo, non sono state utilizzate istruzioni di sprite, quindi la navicella non è ancora uno sprite).
- La riga 47 posiziona le coordinate SSHAPE sulla prima metà (24 per 21 pixel) della capsula della navicella e la memorizza nella stringa dati A$.
- La riga 48 posiziona le coordinate SSHAPE sulla seconda metà (24 x 21 pixel) della navicella e la memorizza nella stringa dati B$.
- La riga 50 trasferisce i dati da A$ allo sprite 1.
- La riga 55 trasferisce i dati da B$ allo sprite 2.
- La riga 60 attiva lo sprite 1 e lo colora di rosso.
- La riga 65 attiva lo sprite 2 e lo colora di blu.
- La riga 82 posiziona lo sprite 1 alla coordinata 150,150.
- La riga 83 posiziona lo sprite 2 a destra di 24 pixel dalla coordinata iniziale dello sprite 1.
- Le righe 82 e 83 praticamente collegano i due sprite.
- Le righe 85 e 87 muovono sullo schermo gli sprite collegati.
- La riga 90 ritarda il programma. Questa volta il ritardo è necessario per poter permettere ai due sprite di completare l'attraversamento dello schermo per due volte. Omettendo questo ritardo gli sprite non avranno tempo sufficiente per spostarsi sullo schermo.
- Le righe 92 e 93 posizionano gli sprite al centro dello schermo e preparano la navicella all'accensione dei razzi posteriori.
- La riga 95 spinge in avanti la capsula (lo sprite 1). Il numero 10 nella riga 95 specifica la velocità dello sprite. La velocità va dal valore 1, che è la posizione di arresto al valore 15 che è la velocità massima.
- La riga 96 fa staccare i razzi posteriori dalla navicella e li fa uscire dallo schermo.

- La riga 97 è un altro ritardo per permettere ai razzi posteriori, lo sprite 2, di uscire dallo schermo.
- La riga 98 disattiva lo sprite 2 quando questo esce dallo schermo.
- La riga 99 è un altro ritardo per permettere alla capsula di continuare a muoversi sullo schermo.
- La riga 100 riporta alla modalità testo.

L'uso di sprite collegati può risultare più interessante di quello di un singolo sprite. I punti più importanti da ricordare sono: (1) Assicurarsi di aver posizionato le coordinate SSHAPE ai punti esatti dello schermo in modo da salvare correttamente i dati della figura; e (2) assicurarsi di aver posizionato correttamente le coordinate quando si effettua l'operazione di collegamento degli sprite con l'istruzione MOVSPR. In questo esempio si era posizionato lo sprite 2 a 24 pixel a destra rispetto allo sprite 1.

Dopo aver appreso come collegare due sprite, provare a collegarne più di due. Più sprite vengono utilizzati, più particolari si potranno aggiungere alle proprie figure.

Il C128 possiede due comandi addizionali SPRITE, SPRCOLOR e COLLISION, che non vengono trattati in questo capitolo. Per informazioni su questi comandi, consultare l'Enciclopedia BASIC 7.0 nel Capitolo V.

Memorizzazione di dati di sprite in file binari

NOTA: La spiegazione che segue presuppone che si sia a conoscenza di linguaggio macchina, locazioni di memoria, file binari e file di codice oggetto.

Il Commodore 128 possiede due nuovi comandi, BLOAD e BSAVE che permettono una gestione chiara e semplice dei dati di sprite. La "B" in BLOAD e BSAVE sta per BINARIO. I comandi BLOAD e BSAVE salvano e caricano file binari da e su disco. Un file binario consiste di una parte di un programma in linguaggio macchina oppure di un insieme di dati all'interno di una gamma di indirizzi specificati. Probabilmente si conoscerà il comando SAVE del monitor del linguaggio macchina incorporato. Quando si utilizza questo comando SAVE, il file risultante sul disco viene considerato un file binario. Un file binario può essere utilizzato con maggiore facilità di un file di codice oggetto in quanto il file binario può essere caricato senza una preparazione. Invece un file di codice oggetto deve essere caricato con un caricatore, come avviene nel Sistema di Sviluppo Assembler del Commodore 64,

quindi per poter eseguire questo file occorre utilizzare il comando sistema (SYS). Per caricare file binari, battere una delle due istruzioni riportate di seguito:

LOAD "nomefile binario",8,1

oppure

BLOAD"nomefile binario",B0,Pinizio

dove "inizio" è 3584 per il caricamento di file di dati di sprite.

Utilizzando il primo metodo si deve specificare alla fine ",1", altrimenti il computer lo caricherà all'inizio del testo BASIC come se si trattasse di un programma BASIC. Il numero ",1" ordina al computer di caricare il file binario nello stesso punto in cui era stato memorizzato.

Per capire cosa ha a che vedere con gli sprite quello che è stato detto, occorre notare che il Commodore 128 ha una porzione di memoria dedicata che va dall'indirizzo decimale 3584 ($0E00) fino a 4095 ($0FFF), nella quale vengono memorizzati i dati di sprite. Questa porzione di memoria occupa 512 byte. Come già detto, uno sprite è composto da 24 pixel di larghezza per 21 pixel di altezza. Ogni pixel richiede un bit di memoria. Se il bit in uno sprite è off (uguale a 0), il pixel corrispondente sullo schermo viene anch'esso considerato off e viene colorato con il colore dello sfondo. Se un bit in uno sprite è on (uguale a 1), il pixel corrispondente sullo schermo sarà anch'esso attivato nel colore del primo piano. La combinazione di "0" e "1" produce l'immagine visualizzata sullo schermo.

Essendo lo sprite di 24 pixel per 21 pixel e richiedendo ogni pixel un bit di memorizzazione nella memoria, uno sprite utilizza 63 byte di memoria. Vedere la Figura 6-8 per comprendere quanto spazio di memorizzazione richiedono i dati di uno sprite.

	12345678	12345678	12345678
1	....	....	....
2	....	....	....
3	....	....	....
4	....	....	....
5	....	....	....
6	....	....	....
7	....	....	....
8	....	....	....
9	....	....	....
10	....	....	....
11	....	....	....
12	....	....	....
13	....	....	....
14	....	....	....
15	....	....	....
16	....	....	....
17	....	....	....
18	....	....	....
19	....	....	....
20	....	....	....
21	....	....	....

Per riga = 24 bit = 3 byte

Figura 6-8. Memoria richiesta dai dati di sprite

Uno sprite richiede 63 byte di dati. Ogni blocco di sprite è composto da 64 byte; il byte addizionale non viene utilizzato. Avendo il Commodore 128 otto sprite ed essendo ognuno di questi sprite composto da un blocco di sprite di 64 byte, il computer richiede 512 (8 x 64) byte per poter rappresentare i dati di tutte le otto figure.

L'area completa in cui sono contenuti gli otto blocchi di sprite inizia alla locazione di memoria 3584 ($0E00) e termina alla locazione 4095 ($0FFF). La figura 6-9 elenca le fasce di valori degli indirizzi di memoria in cui ogni sprite memorizza i propri dati.

```
$0FFF (4095 Decimale)
           ] - Sprite 8
$0FC0
           ] - Sprite 7
$0F80
           ] - Sprite 6
$0F40
           ] - Sprite 5
$0F00
           ] - Sprite 4
$0EC0
           ] - Sprite 3
$0E80
           ] - Sprite 2
$0E40
           ] - Sprite 1
$0E00 (3584 Decimale)
```

Figura 6-9. Fascia di valori degli indirizzi di memoria per la memorizzazione degli sprite

BSAVE

Dopo essere usciti dalla modalità SPRDEF, si potranno salvare i dati nei file binari di sprite. In questo modo è possibile caricare nuovamente un insieme di blocchi di sprite nel Commodore 128 in modo semplice e senza problemi. Per salvare i dati di sprite in un file binario utilizzare il seguente comando:

BSAVE "nomefile binario", B0, P3584 TO P4096

"B0" specifica che si stanno salvando i dati di sprite dal banco 0. I parametri "P3584 TO P4096" significano che si stanno salvando gli indirizzi da 3584 ($0E00) a 4095 ($0FFF) che è la gamma in cui sono memorizzati tutti i dati di sprite.

Non è necessario definire tutti gli sprite quando questi vengono salvati utilizzando il comando BSAVE. Gli sprite definiti infatti vengono salvati (con BSAVE) dal blocco di sprite corretto. Gli sprite non definiti vengono anch'essi salvati nel file binario dal blocco di sprite appropriato, ma non vengono presi in considerazione dal computer. L'operazione di salvataggio (BSAVE) di tutti i 512 byte degli otto sprite, indipendentemente dal fatto che siano utilizzati o meno tutti gli sprite, risulta molto più semplice di quella di salvataggio dei blocchi di sprite individuali.

BLOAD

D'ora in poi, per usare ancora gli sprite, caricare i 512 byte di tutti gli sprite nella gamma di valori che inizia con 3584 ($0E00) e che termina con 4095 ($0FFF) battendo il comando seguente:

BLOAD "nomefile binario"[, B0, P3584]

Utilizzare lo stesso nome di file usato in fase di salvataggio dei dati di sprite originali. "B0" significa banco numero 0 e P3584 specifica la posizione di partenza dove il file binario di dati di sprite viene caricato. Gli ultimi due parametri sono opzionali.

****

In questa sezione si è visto come i nuovi comandi BASIC 7.0 Commodore possano semplificare il processo di creazione di animazione di immagini grafiche che molto spesso risulta complicato. La prossima sezione descrive altri nuovi comandi BASIC 7.0 che rendono semplice la creazione di musica e suoni.

SEZIONE 7

Introduzione

Il Commodore 128 possiede uno dei più sofisticati sintetizzatori di suono incorporati per microcomputer. Il sintetizzatore, chiamato Sound Interface Device (SID) è un chip dedicato esclusivamente alla generazione di suono e musica. Il SID può produrre simultaneamente tre **voci** (suoni) differenti. Ciascuna voce può essere suonata in uno dei quattro tipi di suoni, chiamati forme d'onda. Il SID possiede inoltre i parametri programmabili Attacco, Decadimento, Sostegno e Rilascio (ADSR). Questi parametri definiscono la qualità di un suono. Inoltre, il sintetizzatore è provvisto di un filtro che può essere utilizzato per la scelta di suoni particolari, per eliminare dei suoni o modificare il carattere di un suono o di più suoni. In questa sezione si apprenderà come controllare questi parametri per produrre qualsiasi tipo di suono.

Per facilitare la selezione e l'utilizzo delle molte funzioni del SID, la Commodore ha sviluppato nuove e potenti istruzioni musicali BASIC.

Nel Commodore 128 sono disponibili le seguenti nuove istruzioni per suono e musica:

SOUND
ENVELOPE
VOL
TEMPO
PLAY
FILTER

Le tre istruzioni sonore vengono descritte una per una in questa sezione, nel corso della quale verrà creato un esempio di programma musicale. Alla fine di questa sezione si sarà appreso come comporre un programma musicale. Inoltre si potrà espandere l'esempio fornito e scrivere programmi per suonare composizioni musicali complicate. Si potranno programmare i propri spartiti musicali, creare effetti speciali e suonare le musiche di grande maestri come Beethoven e di artisti contemporanei come i Beatles. Inoltre si potrà aggiungere una colonna sonora ai programmi grafici per creare i propri video.

L'istruzione SOUND

L'istruzione SOUND è stata creata per rendere semplice e veloce la creazione di effetti speciali nei programmi. Più avanti in questa sezione si apprenderà come suonare arrangiamenti musicali completi utilizzando altre istruzioni sonore.

Il formato dell'istruzione SOUND è il seguente:

SOUND VC, FREQ, DUR[, DIR[, MIN[, GL[, FO[, LA]]]]]

Il significato dei parametri è il seguente:

VC	- Seleziona la voce 1, 2 o 3
FREQ	- Imposta il livello di frequenza del suono (da 0 a 65535)
DUR	- Imposta la durata del suono (in sessantesimi di secondo)
DIR	- Imposta la direzione verso cui il suono è incrementato/decrementato 0 = Incrementa la frequenza 1 = Decrementa la frequenza 2 = Provoca un'oscillazione della frequenza.
MIN	- Seleziona la frequenza minima (da 0 a 65535) se viene specificato il glissato (DIR)
GL	- Sceglie il valore di passo del glissato (da 0 a 32767)
FO	- Seleziona la forma d'onda (da 0 a 3) 0 = Triangolo 1 = Dente di sega 2 = Impulso variabile 3 = Rumore bianco
LI	- Imposta la larghezza dell'impulso, la larghezza della forma d'onda ad impulso variabile

Notare che i parametri DIR, MIN, GL, FO, e LI sono opzionali.

Il primo parametro (VC) nell'istruzione SOUND seleziona la voce che sarà suonata. Il secondo parametro (FREQ) determina la frequenza del suono, che va dal valore 0 al valore 65535. La terza impostazione (DUR) specifica la durata del suono. La durata viene misurata in sessantesimi di secondo. Per avere un suono che duri un secondo, impostare la durata a 60 in quanto 1/60 per 60 volte è uguale a 1. Per un suono di due secondi, impostare la durata su 120, per un suono di 10 secondi, impostarla su 600 e così via.

Il quarto parametro (DIR) seleziona la direzione dell'incremento o decremento della frequenza del suono. Questa funzione viene chiamata glissato. La quinta impostazione (MIN) imposta la frequenza minima sul punto in cui inizia il glissato. La sesta impostazione (GL) è il valore di passo del glissato ed è simile al valore di passo in un loop FOR...NEXT. Se vengono specificati i valori DIR, MIN e GL nel comando SOUND, il suono verrà per prima cosa suonato al livello originale specificato dal parametro FREQ.

A questo punto il sintetizzatore glissa e suona ogni livello dell'intera gamma dei valori di frequenza a partire dalla frequenza MIN. Il glissato viene incrementato o decrementato dal valore di passo, a seconda della direzione specificata dal parametro DIR, quindi la frequenza viene suonata al nuovo livello.

Il settimo parametro (FO) nel comando SOUND seleziona la forma d'onda del suono (le forme d'onda vengono trattate dettagliatamente nel paragrafo intitolato **Suono e musica avanzati in modalità C128**).

L'ultima impostazione nel comando SOUND determina la larghezza dell'impulso della forma d'onda se selezionata come il parametro della forma d'onda (consultare la parte relativa al **suono avanzato** per dettagli sulla forma d'onda della larghezza dell'impulso).

Impostazione di un programma SOUND

Si vedrà ora come scrivere il primo programma SOUND. Di seguito viene dato un esempio dell'istruzione SOUND:

```
10 VOL 5
20 SOUND 1,4096,60
```

Eseguire questo programma. Il Commodore 128 emette un suono breve e acuto. Prima di eseguire l'istruzione sonora occorre impostare il volume. Questa è la ragione dell'istruzione nella riga 10, che imposta il volume del chip sonoro. La riga 20 suona la voce 1 ad una frequenza di 4096 per la durata di 1 secondo (1/60 per 60 volte). Modificare la frequenza battendo la seguente istruzione:

```
30 SOUND 1,8192,60
```

Notare che la riga 30 emette un tono più alto di quello della riga 20. Questo mostra la relazione diretta tra l'impostazione della frequenza e l'effettiva frequenza del suono. Aumentando la frequenza il Commodore 128 incrementerà l'altezza del tono. Battere ora la seguente istruzione:

40 SOUND 1, 0, 60

Questo dimostra che un valore FREQ 0 emette la frequenza più bassa (che è talmente bassa da non essere udibile). Un valore FREQ di 65535 emette la più alta frequenza possibile.

Ora includere l'istruzione sonora in un loop FOR...NEXT. Questo permette di suonare la gamma completa di frequenze all'interno del loop. Aggiungere le seguenti istruzioni al programma:

```
50 FOR I = 1 TO 65535 STEP 100
60 SOUND 1,I,1
70 NEXT
```

Questa parte di programma suona la forma d'onda dell'impulso variabile nella gamma di frequenze da 1 a 65535, con incrementi di 100, a partire dalla frequenza più bassa fino alla più alta. Non specificando la forma d'onda, il computer selezionerà il valore di default della forma d'onda 2, la forma d'onda dell'impulso variabile.

Ora modificare la forma d'onda battendo la seguente riga di programma (60) ed eseguire nuovamente il programma:

```
60 SOUND 1,I,1,0,0,0,0,0
```

Ora il programma suona la voce 1 utilizzando la forma d'onda a triangolo, per la gamma di frequenza da 1 a 65535 con incrementi di 100. Questo provoca un suono simile a quello dei video game delle sale giochi. Provare la forma d'onda 1, la forma d'onda a dente di sega ed osservare il suono emesso dopo l'esecuzione della seguente istruzione:

```
60 SOUND 1,I,1,0,0,0,1,0
```

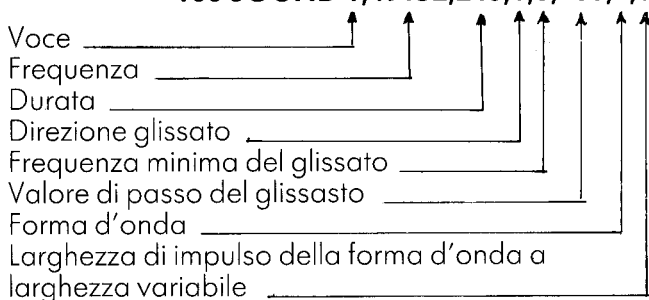
La forma d'onda a dente di sega ha un suono simile a quello della forma d'onda a triangolo sebbene abbia meno ronzii. Infine, provare la forma d'onda del rumore bianco (3). Sostituire la riga 60 precedente con quella riportata di seguito:

```
60 SOUND 1,I,1,0,0,0,3,0
```

Ora il loop di programma suona il generatore di rumore bianco per l'intera gamma di frequenze. Per prima cosa verrà emesso un rombo e, mano a mano che la frequenza aumenta, aumenta l'altezza del tono fino a riprodurre il rumore della partenza di un razzo.

Notare che fino ad ora non sono stati specificati tutti i parametri dell'istruzione SOUND. Ad esempio nella riga 60:

60 SOUND 1,I,1,0,0,0,3,0

I tre 0 dopo 1,I,1 appartengono ai parametri del glissato nell'istruzione SOUND. Siccome non è stato specificato alcun parametro, il suono non glisserà. Aggiungere la seguente riga al programma:

La riga 100 avvia la frequenza del glissato a 49152 e decrementa per centinaia il glissato fino a raggiungere la frequenza minima del glissato (0). La voce 1, che usa la forma d'onda a dente di sega (#1), emette ciascun suono per una durata di quattro secondi (240 * 1/60 di secondo). La riga 100 emette il suono di una bomba che cade come in molti video game delle sale gioco.

Ora modificare alcuni parametri della riga 100. Ad esempio, impostare la direzione del glissato su 2 (oscillazioni); modificare la frequenza minima del glissato a 32768 ed aumentare il valore di passo a 3000. Il comando sarà:

110 SOUND 1, 49152, 240, 2, 32768, 3000, 1

La riga 110 emette un suono di sirena come se la polizia fosse nelle vicinanze. Per ottenere un suono più piacevole battere:

110 SOUND 1, 65535, 250, 0, 32768, 3000, 2, 2600

Il suono sarà quello di armi spaziali che sparano agli alieni.

Fino ad ora si è programmato in una sola voce, però è possibile produrre effetti sonori interessanti con l'istruzione SOUND utilizzando fino a tre voci. Creare ora un programma che utilizzi le tre voci.

Di seguito viene dato un esempio di programma che mostra come programmare il chip del sintetizzatore del Commodore 128. Il programma, all'esecuzione, chiede l'introduzione di alcuni parametri e quindi emette il suono. Il listato del programma è riportato di seguito. Battere il programma ed eseguirlo.

```
10 SCNCLR:PRINT "           SOUND PLAYER":PRINT:PRINT:PRINT
20 PRINT "    INPUT SOUND PARAMETERS TO PLAY":PRINT:PRINT
30 INPUT "VOICE (1-3)";V
40 INPUT "FREQUENCY (0-65535)";F
50 INPUT "DURATION (0-32767)";D:PRINT
60 INPUT "WANT TO SPECIFY OPTIONAL PARAMETERS Y/N";B$:PRINT
70 IF B$="N" THEN 130
80 INPUT "SWEEP DIRECTION (0=UP,1=DOWN,2=OSCILL)";DIR
90 INPUT "MINIMUM SWEEP FREQUENCY (0-65535)";M
100 INPUT "SWEEP STEP VALUE (0-32767)";S
110 INPUT "WAVEFORM (0=TRI,1=SAW,2=VAR PUL,3=NOISE)";W
120 IF W=2 THEN INPUT "PULSE WIDTH (0-4095)";P
130 SOUND V,F,D,DIR,M,S,W,P
140 INPUT "DO YOU WANT TO HEAR THE SOUND AGAIN Y/N";A$
150 IF A$="Y" THEN 130
160 RUN
```

La spiegazione del programma è la seguente. Le righe 10 e 20 visualizzano sullo schermo i messaggi iniziali. Le righe da 30 a 50 introducono i parametri di voce, frequenza e durata.

La riga 60 chiede se si vogliono introdurre i parametri SOUND opzionali, come ad esempio le impostazioni di glissato e la forma d'onda. Se non si desidera specificare alcun parametro, premere il tasto "N". Il programma salterà alla riga 130 ed emetterà il suono. Se invece si desidera specificare l'impostazione SOUND opzionale, premere il tasto "Y". Il programma continuerà l'esecuzione dalla riga 80. Le righe da 80 a 110 specificano la direzione del glissato, la frequenza minima del glissato, il valore di passo del glissato e la forma d'onda. La riga 120 introduce la larghezza dell'impulso della forma d'onda ad impulso variabile solo se viene selezionata la forma d'onda 2 (impulso variabile). Infine, la riga 130 emette il suono a seconda dei parametri specificati precedentemente nel programma.

La riga 140 chiede se si desidera risentire il suono. Se la risposta è affermativa, premere il tasto "Y", se negativa, premere il tasto "N". La riga 150 controlla se si è premuto Y; in caso affermativo il controllo di programma viene ritornato alla riga 130 ed il pro-

gramma emette nuovamente il suono. Non premendo il tasto "Y" il programma continua l'esecuzione della riga 160, ed il programma viene ripetuto. Per arrestare il programma, premere contemporaneamente i tasti RUN/STOP e RESTORE.

Suoni casuali

Il programma seguente genera suoni casuali per mezzo della funzione RND. Ogni parametro di SOUND viene calcolato casualmente. Battere il programma nel computer, salvarlo ed eseguirlo. Questo programma mostra le migliaia di suoni che possono essere prodotte specificando le varie combinazioni dei parametri SOUND. Il listato del programma è il seguente:

```
10 PRINT "VC  FRQ  DIR  MIN   SV   WF   PW ":VOL 5
20 PRINT "-------------------------------------"
30 V=INT (RND (1)*3)+1                  :REM VOICE
40 F=INT (RND (1)*65535)                :REM FREQUENCY
50 D=INT (RND (1)*240)                  :REM DURATION
60 DIR=INT (RND (1)*3)                  :REM STEP DIRECTION
70 M=INT (RND (1)*65535)                :REM MINIMUM FREQUENCY
80 S=INT (RND (1)*32767)                :REM STEP VALUE
90 W=INT (RND (1)*4)                    :REM WAVEFORM
100 P=INT (RND (1)*4095)                :REM PULSE WIDTH
110 PRINT V;F;DIR;M;S;W;P:PRINT:PRINT   :REM DISPLAY VALUES
120 SOUND V,F,D,DIR,M,S,W,P             :REM PLAY SOUND
130 SLEEP 4                             :REM WAIT A BIT
140 SOUND V,0,0,DIR,0,0,W,P             :REM SWITCH SOUND OFF
150 GOTO 10
```

Le righe 10 e 20 stampano le intestazioni di colonna del parametro e la sottolineatura. Le righe da 30 a 100 calcolano ogni parametro sonoro all'interno di una gamma specifica. Ad esempio, la riga 30 calcola il numero di voce come segue:

30 V = INT(RND(1)*3) + 1

La notazione RND (1) specifica il valore **seme** del numero casuale. Il **seme** è il numero di partenza generato dal computer. Il valore 1 ordina al computer di generare un nuovo seme ogni volta che viene incontrato il comando. Avendo il Commodore 128 tre voci, la notazione * 3 ordina al computer di generare un numero casuale nella gamma da 0 a 3. Notare comunque che non esiste la voce 0, quindi il valore + 1 nella riga 30 ordina al computer di generare un numero casuale in modo che 1<=Numero <4. La pro-

cedura di generazione di un numero casuale in una gamma specifica è di moltiplicare il numero casuale specificato per il valore massimo del parametro (in questo caso 3). Se il valore minimo del parametro è maggiore di zero, aggiungere al numero casuale un valore che specificherà il valore minimo della gamma di numeri da generare (in questo caso 1). Ad esempio, la riga 40 genera un numero casuale dato da 0<=Numero <65535. In questo caso il valore minimo è zero, quindi non è necessario aggiungere un valore al numero casuale generato.

La riga 110 stampa i valori dei parametri. La riga 120 emette il suono specificato dai numeri casuali generati nelle righe da 30 a 100. La riga 130 ritarda il programma di 4 secondi durante i quali viene emesso il suono. La riga 140 arresta il suono dopo il ritardo di 4 secondi. Tutti i suoni generati con questo programma saranno per 4 secondi o meno in quanto vengono arrestati dall'istruzione nella riga 140 dopo 4 secondi. Infine, la riga 150 ridà il controllo alla riga 10 ed il processo viene ripetuto fino a quando non vengono premuti contemporaneamente i tasti RUN/STOP e RESTORE.

Fino ad ora sono stati eseguiti programmi di esempio utilizzando solo l'istruzione SOUND. L'istruzione SOUND può essere utilizzata per suonare pezzi musicali di uno spartito, ma la sua funzione principale è quella di generare con facilità e rapidità effetti sonori. Il Commodore 128 possiede istruzioni specifiche per l'esecuzione di pezzi musicali. I paragrafi seguenti descrivono le istruzioni avanzate per suono e musica che permettono di suonare spartiti musicali ed arrangiamenti complessi con il sintetizzatore del Commodore 128.

Suono e musica avanzati in modalità C128

Una breve spiegazione delle caratteristiche del suono

Ogni suono è in effetti un'onda sonora nell'aria. Come una qualsiasi onda, l'onda sonora (sinusoidale) può essere rappresentata graficamente e matematicamente (vedere la Figura 7-1).

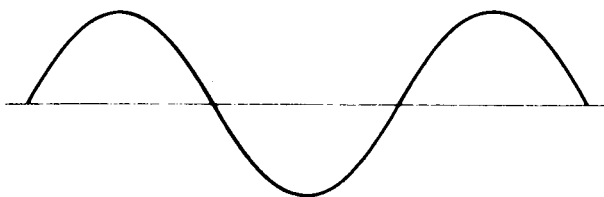



Figura 7-1. Onda sinusoidale

L'onda sonora si muove (oscilla) ad una data velocità (frequenza) la quale determina l'altezza totale del suono (acuto o basso).

Il suono è inoltre composto da armoniche, che sono onde multiple della frequenza totale del suono o della nota. La combinazione di queste onde sonore armoniche dà la qualità della nota, chiamata timbro. La Figura 7-2 mostra la relazione tra le frequenze sonore di base e le armoniche.

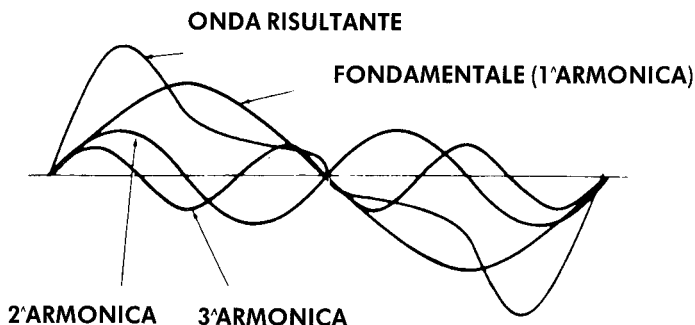



Figura 7-2. Frequenza ed armoniche

Il timbro di un tono musicale (cioè il suono di un tono), viene determinato dalla forma d'onda del tono. Il Commodore 128 può generare quattro tipi di forme d'onda: triangolo, dente di sega, impulso variabile e rumore. Per avere una rappresentazione grafica di queste quattro forme d'onda, vedere la Figura 7-3.

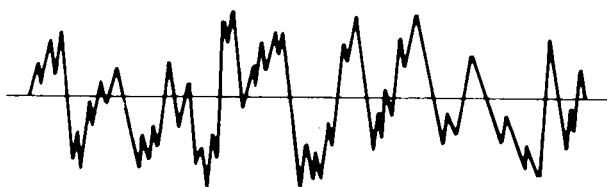



Figura 7-3. Tipi di forme d'onda sonore

L'istruzione ENVELOPE

Il volume di un suono varia per tutta la durata della nota da quando si comincia a sentirla fino a quando non è più udibile. Queste qualità del volume vengono chiamate Attacco, Decadimento, Sostegno e Rilascio (ADSR). L'**attacco** è il momento in cui una nota musicale raggiunge il volume massimo. Il **decadimento** è il momento in cui una nota musicale decresce dal volume massimo fino al livello di sostegno. Il **sostegno** è il livello a cui viene suonata una nota al volume medio. Il **rilascio** è il momento in cui una nota musicale diminuisce dal livello di sostegno fino al volume zero. Il generatore ENVELOPE controlla i parametri ADSR del suono. Per una rappresentazione grafica di ADSR vedere la Figura 7-4. Il Commodore 128 può modificare ogni parametro ADSR fino ad un massimo di 16 diversi valori. Questo permette di avere una grande flessibilità sul generatore ENVELOPE e sulle proprietà del volume alla generazione del suono.

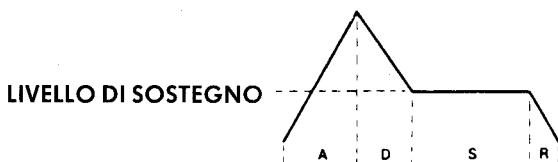



Figura 7-4. Fasi ADSR

Una delle più potenti istruzioni del Commodore 128, quella che controlla ADSR e forma d'onda, è l'istruzione ENVELOPE. Questa istruzione imposta i diversi comandi nel chip del sintetizzatore che rende ciascun suono un suono unico. ENVELOPE permette di gestire il sintetizzatore SID. Con questa istruzione è possibile selezionare una forma d'onda per la musica e gli effetti sonori. Il formato dell'istruzione ENVELOPE è il seguente:

ENVELOPE e[,a[,d[,s[,r[,[,fo[,li]]]]]]]

Il significato di ciascuna lettera è il seguente:

e - numero di inviluppo (da 0 a 9)
a - valore di attacco (da 0 a 15)
d - valore di decadimento (da 0 a 15)
s - livello di sostegno (da 0 a 15)
r - valore di rilascio (da 0 a 15)

fo - forma d'onda - 0 = triangolo
1 = dente di sega
2 = impulso (quadra)
3 = rumore
4 = modulazione ad anello

li - larghezza dell'impulso (da 0 a 4095)

I parametri non ancora definiti vengono spiegati di seguito:

Inviluppo - Le proprietà di una nota musicale specificate dalla forma d'onda e dalle impostazioni di attacco, decadimento, sostegno e rilascio della nota. Ad esempio, l'inviluppo per una nota di chitarra ha un ADSR e una forma d'onda diversi da quelli per un flauto.

Forma d'onda - Il tipo di onda sonora creato dalla combinazione di armoniche di accompagnamento di un tono. Le onde sonore di un'armonica di un'armonica di accompagnamento sono multipli della frequenza totale del tono e si basano su di essa. Le qualità del tono generato da ogni forma d'onda sono diverse una dall'altra e vengono rappresentate graficamente nella Figura 7-3.

Larghezza impulso - L'intervallo tra le note, generato dalla forma d'onda a impulso.

L'istruzione ENVELOPE è molto potente: controlla la maggior parte delle qualità delle note suonate con il sintetizzatore sonoro. Il Commodore 128 possiede 10 inviluppi predefiniti per 10 diversi strumenti musicali. Per utilizzare questi inviluppi non è necessario specificare i parametri ADSR, la forma d'onda e le impostazioni di larghezza di impulso, l'unica cosa da specificare è il numero di inviluppo. Tutti gli altri parametri vengono selezionati automaticamente dal Commodore 128. Di seguito vengono elencati gli inviluppi preselezionati per i diversi tipi di strumenti musicali:

Numero di inviluppo	Strumento	Attacco	Decadimento	Sostegno	Rilascio	Forma d'onda	Larghezza
0	Piano	0	9	0	0	2	1536
1	Fisarmonica	12	0	12	0	1	
2	Calliope	0	0	25	0	0	
3	Tamburo	0	5	5	0	3	
4	Flauto	9	4	4	0	0	
5	Chitarra	0	9	2	1	1	
6	Clavicembalo	0	9	0	0	2	512
7	Organo	0	9	9	0	2	2048
8	Tromba	8	9	4	1	2	512
9	Xilofono	0	9	0	0	0	

Figura 7-5. Parametri di default per l'istruzione ENVELOPE

A questo punto, dopo aver appreso come funziona l'istruzione ENVELOPE, iniziare un altro programma battendo la seguente istruzione:

10 ENVELOPE 0, 5, 9, 2, 2, 2, 1700

Questa istruzione ENVELOPE ridefinisce l'inviluppo di default (0) per il piano nel modo seguente: Attacco = 5, Decadimento = 9, Sostegno = 2, Rilascio = 2, la forma d'onda rimane la stessa (2) e la larghezza dell'impulso della forma d'onda dell'impulso variabile è ora 1700. L'inviluppo per il piano non riceverà queste modifiche finchè non verrà selezionata un'istruzione PLAY che verrà illustrata più avanti in questa sezione.

La prossima fase di programmazione musicale è l'impostazione del volume del chip sonoro nel modo seguente:

20 VOL 8

L'istruzione VOL imposta il volume del chip sonoro ad un valore tra 0 e 15, dove 15 è il valore massimo e 0 è volume disattivato.

L'istruzione TEMPO

La prossima fase di programmazione musicale con il Commodore 128 sarà il controllo del tempo, o velocità del pezzo musicale. Per questa funzione viene utilizzata l'istruzione TEMPO il cui formato è:

TEMPO n

dove n è una cifra compresa tra 0 e 255 (255 è la velocità massima). Non specificando l'istruzione TEMPO in un programma, il Commodore 128 imposterà automaticamente il tempo ad un va-

lore 8. Aggiungere la seguente istruzione all'esempio di programma musicale:

30 TEMPO 10

L'istruzione PLAY

Si vedrà come suonare le note di una canzone. Le note vengono suonate allo stesso modo in cui una stringa di testo viene visualizzata sullo schermo per mezzo del comando PRINT. L'unica differenza è che al posto di PRINT viene utilizzata l'istruzione PLAY. Infatti PRINT visualizza il testo, mentre PLAY emette note musicali.

Il formato dell'istruzione PLAY è il seguente:

PLAY"stringa di caratteri di controllo del sintetizzatore e note musicali"

Il comando PLAY può contenere un massimo di 255 caratteri (comprese note musicali e caratteri di controllo del sintetizzatore). Comunque, siccome questo valore è maggiore del numero massimo di caratteri (160) permessi in una singola riga di programma in BASIC 7.0, per raggiungere questa lunghezza si dovranno concatenare almento due stringhe, oppure, per evitare il concatenamento delle stringhe, occorrerà non superare i 160 caratteri, cioè una riga di programma (questo equivale a quattro righe di schermo se in modalità a 40 colonne e a due righe di schermo se in modalità a 80 colonne). In questo modo si produrranno stringhe di comando PLAY semplici da comprendere e da usare.

Per suonare note musicali, battere tra virgolette le lettere corrispondenti alle note che si desiderano suonare (C=do, D=re, E=mi, F=fa, G=sol, A=la, B=si). Ad esempio, per suonare la scala musicale battere:

40 PLAY "C D E F G A B"

In questo modo verranno suonate le note do, re, mi, fa, sol, la, si nell'inviluppo del piano, cioè l'inviluppo 0. Dopo ogni esecuzione dell'esempio di programma che si sta creando, tenere premuto il tasto RUN/STOP e premere il tasto RESTORE per reimpostare il chip del sintetizzatore.

Ora è possibile specificare la durata della nota facendo precedere la nota stessa da una delle seguenti lettere racchiusa tra vir-

golette:

W - Intero
H - Metà
Q - Quarto
I - Ottavo
S - Sedicesimo

L'impostazione di default, se non viene specificata la durata, sarà W (intero).

É possibile includere una pausa battendo quanto segue nella stringa PLAY:

R - Pausa

Per comunicare al computer di attendere finchè tutte le voci suonate raggiungano la fine di una misura, includere quanto segue racchiuso tra virgolette:

M - attendere la fine della misura

Il Commodore 128 possiede anche caratteri di controllo sintetizzatore che possono essere battuti tra virgolette in una stringa PLAY. Questo permette di avere un controllo completo su ogni nota e di modificare i controlli del sintetizzatore all'interno di una stringa di note. Battere dopo il carattere di controllo un numero compreso nella gamma permessa per quel carattere. I caratteri di controllo e la gamma di numeri per ogni carattere sono mostrati nella Figura 7-6. La lettera "n" dopo il carattere di controllo significa che in quel punto dovrà essere introdotto il numero desiderato.

Carattere di controllo	Descrizione	Gamma	Impostazione di default
V n	Voce	1-3	1
O n	Ottavo	0-6	4
T n	Inviluppo	0-9	0
U n	Volume	0-15	9
X n	Filtro	0 = disattato 1 = attivato	0

Figura 7-6. Caratteri di controllo del sintetizzatore sonoro

Sebbene il chip SID possa elaborare questi caratteri di controllo in qualsiasi ordine, è consigliabile introdurre i caratteri di controllo all'interno di una stringa nello stesso ordine mostrato nella Figura 7-6.

Non è assolutamente necessario specificare i caratteri di controllo, ma sarebbe consigliabile farlo per potenziare le possibilità del sintetizzatore. Il Commodore 128 imposta automaticamente i controlli del sintetizzatore ai valori di default mostrati nella Figura 7-6. Non assegnando i caratteri speciali di controllo, il chip SID potrà suonare un solo inviluppo, una voce e un ottavo senza utilizzare il filtro. Specificare questi caratteri per ottenere il massimo controllo sulle note all'interno della stringa PLAY.

Se si specifica un'istruzione ENVELOPE e si selezionano le impostazioni desiderate invece di utilizzare i parametri di default della Figura 7-5, il numero del carattere di controllo dell'inviluppo nella stringa PLAY dovrà essere lo stesso di quello del numero di inviluppo nell'istruzione ENVELOPE, questo per fare accettare al computer i parametri desiderati. Se si desidera utilizzare le impostazioni di default dell'inviluppo mostrate nella Figura 7-6, non si dovrà specificare l'istruzione ENVELOPE. In questo caso selezionare semplicemente un numero di inviluppo con il carattere di controllo (T) nell'istruzione PLAY.

Di seguito viene fornito un esempio dell'istruzione PLAY utilizzando i caratteri di controllo del chip SID all'interno di una stringa. Aggiungere questa riga al programma e notare la differenza tra questa istruzione e l'istruzione PLAY della riga 40.

50 PLAY "MV2 O5 T7 U5 X0 C D E F G A B"

Con questa istruzione vengono suonate le stesse note della riga 40, con la differenza che è stata selezionata la voce 2, le note vengono suonate più alte di un'ottava (5) rispetto alle note della riga 40, l'impostazione del volume è su 5 ed il filtro è disattivato. Per ora, lasciare disattivato il filtro. Dopo aver appreso nella prossima sezione come utilizzare il filtro, sarà possibile ritornare su questa riga di programma ed attivarlo per vedere l'effetto che avrà sulle note. Notare che la riga 50 seleziona un nuovo strumento, l'inviluppo dell'organo, con il carattere di controllo T7. Ora il programma utilizza due diversi strumenti in due delle voci indipendenti. Aggiungere questa istruzione per suonare la terza voce:

60 PLAY "MV3 O6 U7 T6 X0 C D E F G A B"

Vediamo ora come la riga 60 controlla il sintetizzatore. V3 seleziona la terza voce, O6 alza di un'ottava (6) la voce 3 rispetto alla voce due, T6 seleziona l'inviluppo del clavicembalo, U7 imposta il volume a 7 e X0 disattiva il filtro per tutte le tre voci. Ora il pro-

gramma suona le tre voci, l'una di un'ottava più alta dell'altra, con tre diversi strumenti: piano, organo e clavicembalo.
Fino ad ora si è utilizzata l'istruzione PLAY per suonare note intere. Aggiungere delle note con una diversa durata aggiungendo alla stringa PLAY dei caratteri di controllo durata, come mostrato di seguito:

70 PLAY "MV2 O6 T0 U7 X0 H C D Q E F I G A S B"

La riga 70 suona la voce 2 nell'ottava 6 ad un volume 7 con l'inviluppo del piano (0) ed il filtro disattivato. Questa istruzione suona le note do (C) e re (D) come metà, mi (E) e fa (F) come quarti, sol (G) e la (A) come ottavi e si (B) come sedicesimi. Notare la differenza tra l'inviluppo del piano nella riga 40 e l'inviluppo ridefinito del piano nella riga 70. La riga 40 dà una nota più simile a quella di un piano rispetto alla riga 70.

È possibile suonare diesis, bemolle e note puntate facendo precedere le note desiderate dai seguenti caratteri tra virgolette:

- Diesis
$ - Bemolle
. - Nota puntata

Una nota puntata è lunga una volta e mezzo una nota non puntata.

Ora aggiungere al programma diesis, bemolle e note puntate battendo:

80 PLAY "MV1 O4 T4 U8 X0 .H C D Q # E F I $ G A .S # B"

La riga 80 suona la voce 1 in ottava 4 ad un volume 8 con l'inviluppo del flauto attivato ed il filtro disattivato. Inoltre suona il do (C) ed il re (D) come metà note puntate, mi (E) e fa (F) come quarti diesis, sol (G) e la (A) come ottavi bemolle e si (B) come sedicesimo diesis puntato. È possibile aggiungere pause (R) nella stringa PLAY in qualsiasi momento.

Negli esempi fino a questo momento il filtro del sintetizzatore sonoro è rimasto disattivato e quindi non si è ancora messa alla prova la potenzialità del filtro stesso. Ora, dopo aver appreso la

maggior parte delle istruzioni di suono e musica e i caratteri di controllo del SID, passare alla sezione seguente nella quale verrà spiegato come migliorare la qualità della musica prodotta utilizzando l'istruzione FILTER.

Il filtro del SID

Dopo aver scelto inviluppo, ADSR, volume e tempo, utilizzare il filtro per perfezionare i suoni sintetizzati. Nel programma, l'istruzione FILTER precederà l'istruzione PLAY. Prima di utilizzare il filtro, assicurarsi di aver ben compreso come generare i suoni. Il chip SID possiede un solo filtro, che verrà applicato a tutte e tre le voci. Le musiche prodotte verranno suonate senza filtro, però per sviluppare la potenzialità del sintetizzatore, si consiglia di usare l'istruzione FILTER che aumenterà la qualità e la chiarezza del suono.

Nel primo paragrafo di questa sezione, riguardante le caratteristiche del suono, il suono è stato definito come un'onda che viaggia (oscilla) nell'aria ad una data velocità, che viene definita frequenza dell'onda. Un'onda sonora è composta da una frequenza totale e da armoniche di accompagnamento, che sono multipli della frequenza totale. Vedere la Figura 7-2. Le armoniche di accompagnamento danno al suono il timbro, cioè le qualità del suono determinate dalla forma d'onda. Il filtro all'interno del chip SID permette di mettere in evidenza ed eliminare le armoniche di una forma d'onda e di modificare il suo timbro.

Il filtro del chip SID filtra il suono in tre modi: passa-basso, passa-banda e passa-alto. Questi tipi di filtro sono addizionabili, cioè se ne può utilizzare più di uno alla volta. Per dettagli consultare la prossima sezione.Il filtro passa-basso taglia le frequenze al di sopra del livello specificato (frequenza di taglio). La frequenza di taglio è la linea che divide le frequenze che verranno suonate da quelle che non verranno suonate. Con il tipo di filtro passa-basso, il chip SID suona tutte le frequenze al di sotto della frequenza di taglio ed elimina le frequenze al di sopra di essa. Come dice il nome, le frequenze basse passano attraverso il filtro, mentre quelle alte vengono tagliate. Il filtro passa-basso produce suoni pieni e omogenei. Vedere la Figura 7-7.

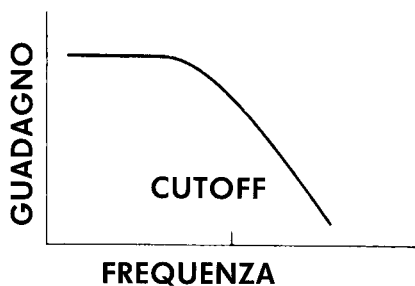



Figura 7-7. Filtro passa-basso

Al contrario, il filtro passa-alto permette a tutte le frequenze al di sopra della frequenza di taglio di passare attraverso il chip. Tutte le frequenze al di sotto di essa vengono tagliate. Vedere la Figura 7-8. Il filtro passa-alto produce suoni metallici e sordi.

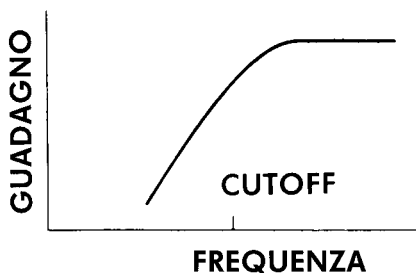



Figura 7-8. Filtro passa-alto

Il filtro passa-banda permette ad una gamma di frequenze al di sopra e al di sotto la frequenza di taglio di passare attraverso il chip SID. Tutte le frequenze al di sopra e al di sotto della banda intorno alla frequenza di taglio vengono eliminate. Vedere la Figura 7-9.

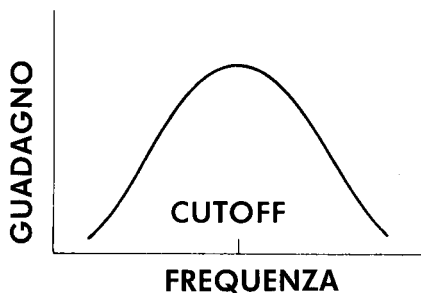



Figura 7-9. Filtro passa-banda

L'istruzione FILTER

L'istruzione FILTER specifica la frequenza di taglio, il tipo di filtro utilizzato e la risonanza. La risonanza è l'effetto di picco della frequenza dell'onda sonora quando si avvicina alla frequenza di taglio. La risonanza determina la nitidezza e la chiarezza di un suono; più la risonanza è alta più il suono è nitido.

Il formato dell'istruzione FILTER è il seguente:

FILTER cf, lp, bp, hp, ris

Il significato dei parametri è il seguente:

freq - Frequenza di taglio (da 0 a 2047)
pb - Filtro passa-basso 0 = disattivato, 1 = attivato
pbd - Filtro passa-banda 0 = disattivato, 1 = attivato
pa - Filtro passa-alto 0 = disattivato, 1 = attivato
ris - Risonanza (da 0 a 15)

La frequenza di taglio può essere compresa in un valore da 0 a 2047. Per attivare il filtro passa-basso, specificare 1 come secondo parametro nell'istruzione FILTER. Per attivare il filtro passa-banda, specificare 1 come terzo parametro e per attivare il filtro passa-alto specificare 1 come quarto parametro. Per disattivare i filtri, introdurre uno 0 nella posizione relativa al filtro da disattivare. È possibile attivare o disattivare uno, due o tutti e tre i filtri contemporaneamente.

A questo punto, dopo aver appreso come usare l'istruzione FILTER, aggiungere la seguente riga al programma sonoro, senza però eseguire il programma.

45 FILTER 1200, 1, 0, 0, 10

La riga 45 imposta la frequenza di taglio a 1200, attiva il filtro passa-basso, disattiva i filtri passa-banda e passa-alto e assegna il valore 10 alla risonanza. Ora attivare il filtro nelle istruzioni PLAY precedenti battendo come carattere di controllo X1 al posto di X0. Reimpostare il chip sonoro premendo i tasti RUN/STOP e RESTORE ed eseguire nuovamente il programma sonoro. Notare la differenza tra il suono delle note ora e quello prodotto senza utilizzare il filtro. Modificare la riga 45 come segue:

45 FILTER 1200, 0, 1, 0, 10

La nuova riga 45 disattiva il filtro passa-basso ed attiva il filtro passa-banda. Premere i tasti RUN/STOP e RESTORE ed eseguire

nuovamente il programma sonoro. Notare la differenza tra il filtro passa-basso ed il filtro passa-banda. Modificare ancora la riga 45 come segue:

45 FILTER 1200, 0, 0, 1, 10

Reimpostare il chip sonoro ed eseguire nuovamente l'esempio di programma. Notare la differenza tra il filtro passa-alto ed i filtri passa-basso e passa-banda. Provare ad utilizzare altre frequenze di taglio, livelli di risonanza e filtri per perfezionare la musica ed il suono dei propri programmi.

Collegamento delle istruzioni del programma musicale

Il vostro primo programma musicale è così completato. Ora è possibile programmare i propri motivi preferiti. Tutti i componenti del programma verranno ora uniti. Il listato del programma sarà quello mostrato di seguito. Sono solamente state aggiunte al programma le istruzioni di stampa in modo da evidenziare le righe di programma che vengono eseguite.

```
10 ENVELOPE 0,5,9,2,2,2,1700
15 VOL 8
20 TEMPO 10
25 PRINT "LINE 30"
30 PLAY "CDEFGAB"
35 FILTER 1200,0,0,1,10
40 PRINT "LINE 45 - FILTER OFF"
45 PLAY "V2 O5 T7 U5 X0 CDEFGAB"
50 PRINT "SAME AS LINE 45 - FILTER ON"
55 PLAY "V2 O5 T7 U5 X1 CDEFGAB"
60 PRINT "LINE 65 - FILTER OFF"
65 PLAY "V3 O6 U7 T6 X0 CDEFGAB"
70 PRINT "SAME AS LINE 65 - FILTER ON"
75 PLAY "V3 O6 U7 T6 X1 CDEFGAB"
80 PRINT "LINE 85 - FILTER OFF"
85 PLAY "V2 O6 T0 U7 X0 HCD QEF IGA SB"
90 PRINT "SAME AS LINE 85 - FILTER ON"
95 PLAY "V2 O6 T0 U7 X1 HCD QEF IGA SB"
100 PRINT "LINE 105 - FILTER OFF"
105 PLAY "V1 O4 T4 U8 X0 H.CD Q#EF I$GA S.B"
110 PRINT"SAME AS LINE 105 - FILTER ON"
115 PLAY "V1 O4 T4 U8 X1 H.CD Q#EF I$GA S.B"
```

Nella riga 10, l'istruzione ENVELOPE specifica l'inviluppo del piano (0) che imposta l'attacco a 0, il decadimento a 9, il sostegno a 0 ed il rilascio a 0. Inoltre questa istruzione seleziona la forma d'onda dell'impulso variabile con una larghezza di impulso 1700. La riga 15 imposta il volume a 8. La riga 20 imposta il tempo a 10.

La riga 35 filtra le note suonate nelle righe da 30 a 115 ed imposta la frequenza di taglio del filtro a 1200. Inoltre, la riga 35 disattiva i filtri passa-basso e passa-banda introducendo due 0 dopo la frequenza di taglio (1200). Il filtro passa-alto viene attivato introducendo un 1 dopo i due 0. La risonanza viene impostata a 10 introducendo l'ultimo parametro nell'istruzione FILTER.

La riga 30 suona le note do, re, mi, fa, sol, la, si (C, D, E, F, G, A, B) in questo ordine. La riga 45 suona le stesse note della riga 30, specificando i caratteri di controllo del SID U5 come livello di volume 5, V1 come voce 1 e O5 come ottava 5. Ricordare che i caratteri di controllo del SID permettono di modificare i controlli del sintetizzatore all'interno di una stringa e di avere il massimo controllo sul sintetizzatore. La riga 65 specifica i caratteri di controllo U7 per il livello di volume 7, V3 per la voce 3, O6 per l'ottava 6 e X0 per la disattivazione del filtro. La riga 65 suona le stesse note delle righe 30 e 45, con volume, voce ed ottava differenti.

La riga 85 ha lo stesso volume, voce ed ottava della riga 65, e specifica metà per le note do (C) e re (D), quarti per le note mi (E) e fa (F), ottavi per le note sol (G) e la (A) e sedicesimi per la nota si (B). La riga 105 imposta il volume a 7, la voce 1, l'ottava 4 e disattiva il filtro. Inoltre la riga 105 specifica che il do (C) è una metà nota puntata, il sol (E) è un quarto diesis, il sol (G) e il la (A) sono ottavi bemolle e il si (B) è un sedicesimo diesis puntato.

Il filtro in programmazioni più avanzate

Negli esempi precedenti si è utilizzato un solo filtro per volta. È però possibile combinare tra di loro i tre filtri del chip SID in modo da ottenere diversi effetti. Ad esempio, è possibile attivare contemporaneamente i filtri passa-basso e passa-alto in modo che formino un filtro di blocco a ripidità di fronte. Questo tipo di filtro permette alle frequenze al di sotto e al di sopra della frequenza di taglio di passare attraverso il chip SID mentre le frequenze vicine alla frequenza di taglio vengono filtrate. Vedere la Figura 7-10 per avere una rappresentazione grafica del filtro di blocco a ripidità di fronte.

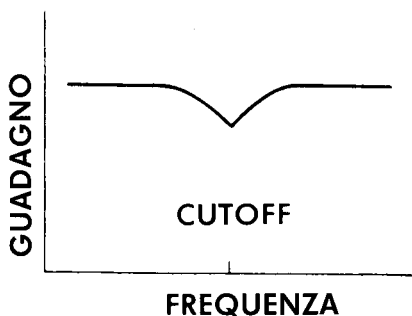



Figura 7-10. Filtro di blocco a ripidità di fronte

Per ottenere effetti particolari è possibile aggiungere al filtro passa-banda sia il filtro passa-basso sia quello passa-alto. Combinando il filtro passa-banda al filtro passa-basso, verrà selezionata la banda di frequenza al di sotto della frequenza di taglio. Le altre frequenze verranno tagliate.

Combinando i filtri passa-banda e passa-alto, verrà selezionata la banda di frequenza al di sopra della frequenza di taglio. Tutte le frequenze al di sotto di essa verranno tagliate.

Provare le diverse combinazioni di filtri per dare diversi accenti alle note musicali e agli effetti sonori. I filtri sono stati studiati per perfezionare i suoni creati dagli altri componenti del chip SID. Dopo aver creato le note musicali o gli effetti sonori con il chip SID, tornare sugli esempi precedenti ed attivare il filtro in modo da ottenere una musica più incisiva e limpida.

A questo punto si sono apprese tutte le informazioni utili per poter scrivere i propri programmi musicali nel linguaggio BASIC del Commodore 128. Utilizzare le diverse forme d'onda, le impostazioni ADSR, le istruzioni TEMPO e FILTER. Consultare un libro di spartiti musicali e battere le note di una scala musicale in sequenza all'interno di una stringa. Accentuare le note nella stringa utilizzando il sintetizzatore musicale del Commodore 128 con la grafica della modalità C128 per creare propri video completi di colonna sonora.

Codifica di un pezzo musicale da uno spartito

Questa sezione fornisce un esempio di spartito musicale e spiega come decodificare le note di un rigo musicale e tradurle in una forma comprensibile per il Commodore 128. Questo esercizio ri-

sulterà più veloce e più semplice per coloro che sanno leggere la musica. Comunque, non è necessario essere un musicista per poter suonare con il Commodore 128. Per coloro che non sanno leggere la musica, la Figura 7-11 mostra un rigo musicale e come le note sul rigo corrispondono ai tasti di un pianoforte.

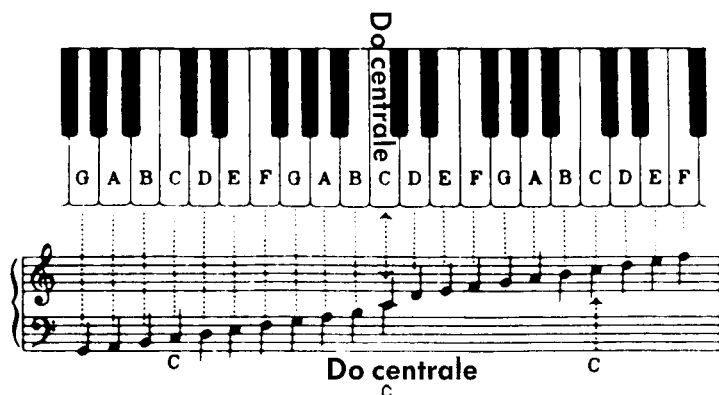



Figura 7-11. Rigo musicale

La Figura 7-12 fornisce un estratto dalla composizione Inventio 13 di Johann Sebastian Bach. Sebbene questa musica sia stata scritta qualche centinaio di anni fa, potrà essere suonata con un sintetizzatore moderno come il chip SID del Commodore 128. Di seguito vengono fornite le misure di apertura dell'Inventio 13.

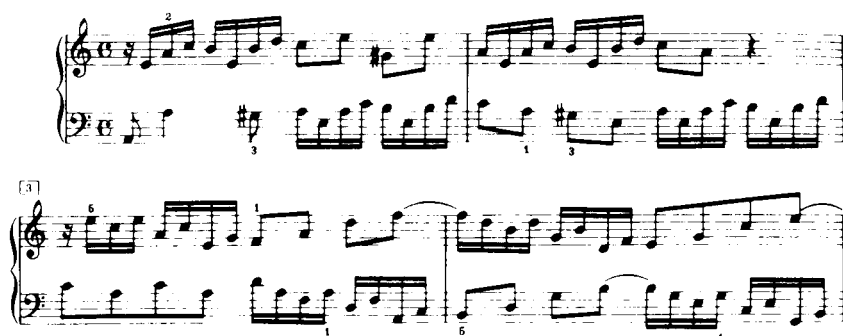



Figura 7-12. Estratto da Inventio 13 di Bach

Il miglior modo per iniziare la codifica di un pezzo musicale con il Commodore 128 è di modificare le note in un codice intermedio. Scrivere le note del primo rigo musicale su un foglio di carta. Ora scrivere le note del rigo sottostante. Davanti ai valori delle note scrivere un codice di durata. Ad esempio, davanti a un ottavo, battere un 8, ad un sedicesimo un 16 e così via. Quindi, separare le note in modo che le note di una misura del rigo superiore siano proporzionali per quanto riguarda il tempo alle note di una misura del rigo sottostante.

Se la composizione musicale avesse un terzo rigo, lo si separerà in modo che la durata sia proporzionale ai due pentagrammi precedenti. Dopo che le note di tutti i pentagrammi saranno separate in durate uguali, una voce dedicata separata suonerà ogni nota di un rigo particolare. Ad esempio, la voce 1 suonerà il primo rigo, la voce 2 il secondo, la 3 il terzo, se esistente.

Supponiamo che il primo pentagramma inizi con una stringa di quattro ottavi ed il secondo rigo con una stringa di otto sedicesimi. Essendo un ottavo proporzionale a due sedicesimi, separare la nota come mostrato nella Figura 7-13.

V1 =	**8A**	**8B**	**8C**	**8D**
V2 =	**16D 16E**	**16F 16G**	**16A 16B**	**16C 16D**

Figura 7-13. Sincronizzazione delle note per due voci

Siccome in una composizione musicale la sincronizzazione ed il tempo sono molto importanti, assicurarsi che le note nel rigo superiore per la voce 1 ad esempio, siano in accordo per quanto riguarda il tempo con le note del rigo sottostante per la voce 2. La prima nota del primo rigo riprodotto nella Figura 7-12 è un ottavo di la (A). Le prime due note per la voce 2 sono sedicesimi di re (D) e mi (E). In questo caso per prima cosa si dovrà introdurre l'ottavo della voce 1 nella stringa PLAY, quindi introdurre dopo di esso il sedicesimo per la voce 2. Per continuare l'esempio, la seconda nota della Figura 7-12 per la voce 1 (rigo superiore) è un ottavo di si (B). L'ottavo di si è uguale come tempo ai due sedicesimi, fa (F) e sol (G), che sono rappresentati nell'ultimo rigo per la voce 2. Per poter coordinare il tempo, introdurre l'ottavo di si (B) nella stringa per la voce 2 seguito dai due sedicesimi, fa (F) e sol (G), per la voce 2.

Iniziare sempre con la nota che avrà la durata più lunga. Ad esempio, se una riga inizia con una serie di due sedicesimi sul rigo inferiore per la voce 2 e il rigo superiore inizia con un ottavo per la voce 1, introdurre nella stringa per prima cosa l'ottavo in quanto dovrà suonare per l'intera durata mentre i due sedicesimi no. Si dovrà dare al computer il tempo sufficiente per poter suonare la nota più lunga e quindi le note più corte, altrimenti la composizione non sarà sincronizzata.

Di seguito viene riportato il programma per suonare Inventio 13. Batterlo e salvarlo per un uso futuro, quindi eseguirlo.

```
10 REM INVENTION 13 BY J S BACH
20 TEMPO 6
30 A$="V1O4T7U8X0 V2O4T7U8X0":REM V1=ORGAN, V2=PIANO
40 DO
50 PLAY A$
60 READ A$
70 LOOP UNTIL A$="END OF MUSIC"
80 END
90 REM **** FIRST MEASURE
100 DATA V2O1IA V1O3IE V2O2QA V1O3SAO4CO3BEM
110 DATA V2O2I#G V1O3SBO4DO4IC V2O2SAEM
120 DATA V1O4IE V2O2SAO3C V1O3I#G V2O2SBEM
130 DATA V1O4IE V2O2SBO3DM
140 REM **** SECOND MEASURE
150 DATA V2O3IC V1O3SAE V2O2IA V1O3SAO4CM
160 DATA V2O2I#G V1O3SBE V2O2IE V1O3SBO4DM
```

```
170 DATA V1O4ICV 2O2SAE V1O3IA V2O2SAO3CM
180 DATA V1O4QR V2O2SBEBO3DM
190 REM **** THIRD MEASURE
200 DATA V2O3IC V1O4SRE V2O2IA V1O4SCEM
210 DATA V2O3IC V1O3SAO4C V2O2IA V1O2SEGM
220 DATA V1O3IF V2O3SDO2A V1O3IA V2O2SFAM
230 DATA V1O4ID V2O2SDF V1O4IF V2O1SAO2CM
240 REM **** FOURTH MEASURE
250 DATA V2O1IB V1O4SFD V2O2ID V1O3SBO4DM
260 DATA V2O2IG V1O3SGB V2O2IB V1O3SDFM
270 DATA V1O3IE V2O2SGE V1O3IG V2O2SEGM
280 DATA V1O4IC V2O2SCE V1O4IE V2O1SGBM
290 REM **** FIFTH MEASURE
300 DATA V2O1IA V1O4SEC V2O2IC V1O3SAO4CM
310 DATA V1O3IF V2O2SDF V1O4ID V2O1SBO2DM
320 DATA V2O1IG V1O3SDB V2O1IB V1O3SGBM
330 DATA V1O3IE V2O2SCE V1O4IC V2O1SAO2CM
340 REM **** SIXTH MEASURE
350 DATA V2O1IF V1O4SCO3A V2O1ID V1O3SFAM
360 DATA V1O3ID V2O1SGO2G V1O3IB V2O2SFGM
370 DATA V2O1IA V1O4SCO3A V2O2I#F V1O4SCEM
380 DATA V2O1IB V1O4SDO3B V2O2I#G V1O4SDFM
390 REM **** SEVENTH MEASURE
400 DATA V2O2IC V1O4SEC V2O2IA V1O4SEGM
410 DATA V2O2ID V1O4SFE V2O2I$B V1O4SDCM
420 DATA V2O2I#G V1O3SBO4C V2O2IF V1O4SDEM
430 DATA V2O2ID V1O4SFD V2O1IB V1O4S#GDM
440 REM **** EIGHTH MEASURE
450 DATA V2O2I#G V1O4SBD V2O2IA V1O4SCAM
460 DATA V2O2ID V1O4SFD V2O2IE V1O3SBO4DM
470 DATA V2O2IF V1O3S#GB V2O2I#D V1O4SCO3AM
480 DATA V2O2IE V1O3SEA V2O2IE V1O3SB#GM
490 REM **** NINTH MEASURE
500 DATA V2O1HA V1O3SAECEO2QAM
510 REM **** END OF MUSIC ****
520 DATA END OF MUSIC
```

Per codificare i pezzi musicali preferiti e suonarli con il Commodore 128 utilizzare le tecniche descritte in questa sezione.

****

Nelle sezioni precedenti si è visto come utilizzare i nuovi e potenti comandi in linguaggio BASIC 7.0 per la modalità C128. Nella sezione seguente si apprenderà come utilizzare le visualizzazioni a 40 e 80 colonne sul Commodore 128.

SEZIONE 8

Introduzione

Nelle modalità C128 e CP/M è possibile utilizzare sia la modalità a 40 colonne di schermo sia quella a 80 colonne. In un solo programma possono essere utilizzate entrambe le modalità.

Ogni dimensione di schermo viene usata per scopi particolari. Lo schermo a 40 colonne è lo stesso tipo di schermo utilizzato dal Commodore 64. Con la visualizzazione a 40 colonne si potranno usare le funzioni grafiche del Commodore 128 per disegnare cerchi, grafici, caratteri di sprite, riquadri ed altre figure in alta risoluzione oppure in modalità grafica multicolore. Inoltre si potranno utilizzare gli sprite.

Con la modalità a 80 colonne si ha a disposizione il doppio di caratteri della modalità a 40 colonne per ogni riga di programma. In modalità a 80 colonne si potranno selezionare per mezzo della tastiera i caratteri grafici ed i colori disponibili.

Per sfruttare appieno le potenzialità di entrambi i formati di schermo si potranno scrivere programmi utilizzando due monitor ognuno dei quali eseguirà parti diverse del programma. Ad esempio, l'output del testo potrebbe essere visualizzato sul monitor a 80 colonne mentre l'output grafico su quello a 40 colonne.

Il tasto 40/80

Per impostare la larghezza di schermo su 40 o 80 colonne, utilizzare il tasto 40/80. Questo tasto avrà effetto solo quando viene intrapresa una delle seguenti azioni:

1. Il computer è stato acceso.
2. Il pulsante RESET è stato premuto.
3. I tasti RUN/STOP e RESTORE sono stati premuti contemporaneamente.

Il tasto 40/80 funziona come un tasto SHIFT/LOCK, cioè viene attivato alla pressione e rimane attivato fino a quando non viene premuto una seconda volta. Premendo questo tasto dopo che si è verificata una delle precedenti tre condizioni, lo schermo verrà attivato sulla modalità a 40 colonne. Se invece viene premuto prima dell'accensione del computer, nonostante si sia verificata una delle condizioni precedenti, lo schermo sarà impostato sulla modalità a 80 colonne. Non è possibile attivare un formato diverso da quello che si sta utilizzando correntemente (40 o 80 co-

lonne) per mezzo del tasto 40/80. Se si dovesse verificare questa necessità, occorre premere e rilasciare il tasto ESC seguito dal tasto X.

Uso del software già pronto a 80 colonne

La maggior parte dei programmi CP/M utilizza una visualizzazione a 80 colonne, come avviene per i pacchetti applicativi da utilizzare in modalità C128. Dal momento che la larghezza di un normale foglio di carta stampata è 80 colonne, un normale word-processor a 80 colonne potrà visualizzare le informazioni sullo schermo esattamente come appariranno sul foglio. Anche i programmi per fogli elettronici specificano spesso il formato a 80 colonne in modo da fornire spazio sufficiente alle diverse colonne e categorie di informazioni. Anche molti pacchetti database e programmi di telecomunicazioni richiedono o possono utilizzare la visualizzazione a 80 colonne.

Creazione di programmi a 80 colonne

Oltre all'esecuzione dei programmi già pronti per l'uso, l'ampiezza di schermo a 80 colonne può risultare utile nella creazione di propri programmi. Si sarà notato che battendo una riga superiore alle 40 colonne su uno schermo a 40 colonne la riga va a capo, cioè continua sulla riga sottostante. Questo potrebbe causare confusione nella lettura della riga e potrebbe anche causare errori di programmazione. La visualizzazione a 80 colonne permette di evitare questo tipo di errori fornendo più spazio per una maggiore chiarezza nelle righe di programma.

Uso contemporaneo di 40 e 80 colonne

Il vantaggio di un output composito a 40 colonne è la possibilità di ottenere grafici a matrice di punti, mentre la visualizzazione a 80 colonne permette di avere l'output per l'elaborazione testi e per altre applicazioni commerciali. Se si possiedono due monitor, è possibile scrivere programmi "condivisi" utilizzando le funzioni di testo fornite dalla visualizzazione a 80 colonne ed i grafici della visualizzazione a 40 colonne. All'interno di un programma può essere usato un comando speciale (GRAPHIC 1,1) per trasferire l'esecuzione dei comandi grafici sulla visualizzazione a 40 colonne. Utilizzando un monitor duale (che visualizza cioè i formati a 40 e a 80 colonne) le istruzioni GRAPHIC 1,1 vengono introdotte nel programma per ottenere grafici nel formato a 40 colonne. Però, per

poter visualizzare questi grafici, si dovrà impostare l'interruttore del monitor su 40 colonne. Quando viene scritto un programma di questo tipo, sarebbe consigliabile includere informazioni che verranno visualizzate sullo schermo per ricordare all'utente di impostare l'interruttore.

Ad esempio, si potrebbe scrivere un programma che chieda all'utente di introdurre i dati, e che quindi crei un diagramma a barre basato sull'input dell'utente. Introducendo il messaggio "PER VISUALIZZARE IL GRAFICO ATTIVARE 40 COLONNE" si ricorderà all'utente di invertire le modalità per visualizzare il risultato.

Come già detto precedentemente, è possibile passare dal formato a 80 colonne a quello a 40 colonne dopo l'accensione del computer utilizzando la sequenza ESC/X.

L'esempio seguente mostra come all'interno di un programma possono essere usati gli schermi duali:

```
10 GRAPHIC 5,1:SCNCLR    :REM SWITCH TO 80 COLUMN AND CLEAR IT.
20 PRINT "START IN 40 COLUMN BY SELECTING THE COMPOSITE VIDEO"
30 PRINT "INPUT OF YOUR DUAL MONITOR."
40 PRINT
50 PRINT "PRESS THE RETURN KEY WHEN READY."
60 GETKEY A$:IF A$ <> CHR$(13) THEN 60
70 GRAPHIC 2,1           :REM SELECT SPLIT SCREEN MODE.
80 CHAR 1,8,18,"BIT MAP/TEXT SPLIT SCREEN"
90 FOR I = 70 TO 220 STEP 20:CIRCLE 1,I,50,30,30:NEXT I
100 PRINT
110 PRINT " SWITCH TO 80 COLUMN BY SELECTING THE
120 PRINT " RGBI VIDEO INPUT OF YOUR DUAL MONITOR,"
130 PRINT " THEN PRESS THE RETURN KEY WHEN READY."
140 GETKEY A$:IF A$ <> CHR$(13) THEN 140
150 GRAPHIC 5,1          :REM SWITCH OUTPUT TO THE 80 COLUMN.
160 FOR J = 1 TO 10
170 PRINT "YOU ARE NOW IN 80 COLOUMN TEXT MODE."
180 NEXT J:PRINT
190 PRINT "NOW SWITCH BACK TO 40 COLUMN OUTPUT.":PRINT
200 PRINT "PRESS THE RETURN KEY WHEN READY."
210 GETKEY A$:IF A$ <> CHR$(13) THEN 210
220 GRAPHIC 0,1          :REM SWITCH OUTPUT TO THE 40 COLUMN.
230 PRINT
240 FOR J = 1 TO 10
250 PRINT " YOU ARE NOW IN 40 COLUMN TEXT OUTPUT."
260 NEXT J
270 END
```

Ogni formato di schermo offre dei vantaggi; comunque i due tipi di visualizzazione possono essere combinati in un programma per completarsi l'un l'altro. La visualizzazione a 40 colonne offre la potenzialità della grafica avanzata BASIC. La visualizzazione a 80 colonne offre maggior spazio per i propri programmi. Inoltre, questa visualizzazione permette di eseguire la vasta gamma di programmi studiati per la modalità a 80 colonne.

****

Le sezioni di questo capitolo hanno trattato le molte caratteristiche e funzioni fornite dal Commodore 128 in modalità C128. Il capitolo seguente mostrerà come utilizzare la modalità C64 del Commodore 128.

CAPITOLO 3

USO DELLA MODALITÀ C64

SEZIONE 9

USO DEL BASIC 2.0

L'intero linguaggio BASIC 2.0 del Commodore 64 è stato incorporato nel linguaggio BASIC 7.0 del Commodore 128. I comandi BASIC 2.0 possono essere utilizzati sia in modalità C128 sia in modalità C64. Per ulteriori dettagli su questi comandi consultare le Sezioni 3 e 4 nel Capitolo II.

Set di caratteri della tastiera

Nell'illustrazione della tastiera fornita nella Sezione 3, i tasti **ombreggiati** sono i tasti da utilizzare in modalità C64. In questa modalità la tastiera ha gli stessi due set di caratteri della modalità C128:

- Set di caratteri maiuscolo/grafico
- Set di caratteri maiuscolo/minuscolo

All'attivazione della modalità C64, la tastiera sarà impostata sul set di caratteri maiuscolo/grafico, quindi tutte le lettere battute saranno maiuscole. In modalità C64 è possibile utilizzare un solo set di caratteri per volta. Per passare da un set di caratteri all'altro, premere il tasto SHIFT contemporaneamente al tasto C= (tasto COMMODORE).

Uso dei tasti come su una macchina per scrivere

Come nella modalità C128, nella modalità C64 è possibile utilizzare i tasti come quelli di una macchina per scrivere per ottenere lettere maiuscole e minuscole. Si possono inoltre battere i tasti numerici sulla fila di tasti in alto alla tastiera principale e battere i simboli grafici rappresentati sul lato frontale dei tasti.

Uso dei tasti di comando

La maggior parte dei tasti di comando (cioè quei tasti che inviano i messaggi al computer, come ad esempio RETURN, SHIFT, CTRL ecc.) funzionano in modalità C64 come in modalità C128.

L'unica differenza è che in modalità C64 il cursore può essere spostato solo utilizzando i due tasti CRSR nell'angolo inferiore destro della tastiera principale (in modalità C128 invece si possono usare i tasti freccia situati in alto a destra sulla tastiera principale).

Spostamento del cursore in modalità C64

In modalità C64, per spostare il cursore vengono utilizzati i tasti CRSR sulla tastiera principale ed il tasto SHIFT, come descritto nella Sezione 3.

Programmazione dei tasti funzione in modalità C64

I quattro tasti situati sul lato destro della tastiera, al di sopra del tastierino numerico, vengono chiamati **tasti funzione** e sono contrassegnati F1, F3, F5 e F7 sul lato superiore e F2, F4, F6 e F8 sul lato frontale. Questi tasti possono essere **programmati** - cioè vengono destinati all'esecuzione di compiti o funzioni specifiche. Per questa ragione questi tasti vengono spesso chiamati **tasti funzione programmabili**.

Per ottenere le funzioni associate con quanto rappresentato sul lato frontale dei tasti - cioè F2, F4 F6 e F8, tenere premuto il tasto SHIFT. Per questo motivo questi tasti vengono spesso chiamati **tasti funzione programmabili con SHIFT.**

Ai tasti funzione della modalità C64 non viene assegnato un carattere stampato. Comunque, a questi tasti vengono assegnati codici CHR$. Effettivamente, ognuno di questi tasti ha due codici CHR$ - uno per quando il tasto viene premuto normalmente e l'altro per quando il tasto viene premuto contemporaneamente a SHIFT. Per ottenere la funzione dei tasti con numerazione pari, tenere premuto il tasto SHIFT contemporaneamente al tasto desiderato. Ad esempio, per attivare F2, premere SHIFT con F1.

I codici CHR$ per i tasti da F1 a F8 vanno da 133 a 140. Comunque, i codici non vengono assegnati ai tasti in ordine numerico, ma come indicato di seguito:

F1 CHR$(133)
F2 CHR$(137)
F3 CHR$(134)
F4 CHR$(138)
F5 CHR$(135)
F6 CHR$(139)
F7 CHR$(136)
F8 CHR$(140)

I tasti funzione possono essere utilizzati in diversi modi all'interno di un programma per mezzo dell'istruzione GET (Vedere la Sezione 4 per dettagli sull'istruzione GET). Ad esempio, il pro-

gramma seguente assegna al tasto F1 la funzione di stampa di un messaggio sullo schermo:

```
10 ?"PREMERE F1 PER CONTINUARE"
20 GET A$:IF A$=""THEN 20
30 IF A$<>CHR$(133) THEN 20
40 ? "HAI PREMUTO F1"
```

Le righe 20 e 30 eseguono la maggior parte del lavoro in questo programma. La riga 20 ordina al computer di attendere fino alla pressione di un tasto prima di continuare l'esecuzione del programma. Notare che quando il comando immediatamente successivo a THEN è un GOTO, è necessario introdurre solo il numero di riga. Notare inoltre che un comando GOTO può essere utilizzato per spostarsi sulla stessa riga in cui è contenuto. La riga 30 ordina al computer di tornare indietro e di attendere la pressione di un altro tasto a meno che non sia stato premuto il tasto F1.

SEZIONE 10

Dopo aver creato un programma, per memorizzarlo definitivamente per poterlo riutilizzare in seguito occorre un disk drive Commodore o il Datassette Commodore.

Formattazione di un disco in modalità C64

Per memorizzare un programma su un disco vuoto o vergine, per prima cosa si dovrà preparare il dischetto in modo che possa ricevere i dati. Questa operazione viene chiamata formattazione del disco. Prima di inserire un disco, assicurarsi di aver acceso il disk drive.

Per formattare un disco nuovo in modalità C64, battere il comando seguente:

OPEN 15,8,15:PRINT#15,"N0:NOME,ID" RETURN

Dove nell'esempio è stato indicato "NOME", battere il nome del disco desiderato che non dovrà superare i 16 caratteri di lunghezza. Dove è stato indicato ID, battere un codice qualsiasi composto da due caratteri (es. W2 o 10).

Nel corso dell'operazione di formattazione il cursore scompare dallo schermo. Quando riappare, battere il seguente comando:

CLOSE 15 RETURN

NOTA: Dopo che un disco è stato formattato in modalità C64 o C128 potrà essere utilizzato in entrambe le modalità.

Il comando SAVE

Il comando SAVE viene utilizzato per memorizzare il programma su disco o cassetta.

Salvataggio su disco

Se si possiede un disk drive singolo Commodore, per memorizzare il programma battere:

SAVE "NOME PROGRAMMA",8 RETURN

Il numero 8 indica al computer che si sta utilizzando un disk drive per la memorizzazione del programma.

Le stesse regole vengono applicate a NOME PROGRAMMA a seconda che si stia utilizzando un disco o una cassetta. Il nome programma può essere un nome qualsiasi, composto da lettere, numeri e/o simboli con una lunghezza massima di 16 caratteri.

Notare che il NOME PROGRAMMA deve essere racchiuso tra virgolette. Durante il salvataggio del programma il cursore scompare dallo schermo e riappare al completamento dell'operazione.

Salvataggio su cassetta

Se si utilizza un Datassette per la memorizzazione del programma, inserire una cassetta vergine nel registratore, riavvolgere completamente il nastro (se necessario) e battere:

SAVE "NOME PROGRAMMA" RETURN

I comandi LOAD e RUN

Dopo il salvataggio di un programma, sarà possibile ricaricarlo nella memoria del computer ed eseguirlo ogni volta che sarà necessario.

Caricamento ed esecuzione del programma da disco

Per caricare un programma da disco, battere:

LOAD"NOME PROGRAMMA",8 RETURN

Anche in questo caso il numero 8 indica al computer che si sta lavorando con un disk drive.

Per eseguire il programma, battere RUN e premere <**RETURN**>.

Caricamento ed esecuzione del programma da cassetta

Per caricare il programma da cassetta, battere:

LOAD "NOME PROGRAMMA" RETURN

Se non si conosce il nome del programma, battere:

LOAD RETURN

In questo modo verrà caricato il prossimo programma sulla cassetta.

Per identificare il punto di partenza di un programma è possibile utilizzare il contatore del Datassette. Quindi per richiamare un programma, portare il nastro dalla posizione 000 al punto di inizio del programma e battere:

LOAD RETURN

In questo caso non si dovrà specificare il NOME PROGRAMMA; il programma infatti sarà caricato automaticamente in quanto si tratterà del primo programma presente sulla cassetta.

NOTA: Durante il processo di caricamento, il programma caricato non viene cancellato dal nastro, ma semplicemente copiato nel computer. Comunque, il processo di caricamento di un programma causerà automaticamente la cancellazione di qualsiasi programma BASIC presente nella memoria del computer.

Altri comandi disco

Verifica di un programma

Per verificare il corretto salvataggio o caricamento di un programma, battere:

VERIFY"NOME PROGRAMMA",8 RETURN

Se il programma nella memoria del computer è identico a quello su disco, lo schermo visualizzerà: "OK".

Il comando VERIFY funziona anche per i programmi su cassetta. Battere:

VERIFY "NOME PROGRAMMA" RETURN

Notare che non è necessario battere la virgola ed il numero 8 in quanto 8 indica che si sta lavorando su un disk drive.

Visualizzazione dell'elenco del disco

Per visualizzare la lista dei programmi su disco, per prima cosa battere:

LOAD"$",8 RETURN

Durante questo processo il cursore scomparirà. Quando il cursore riappare, battere:

LIST RETURN

A questo punto verrà visualizzata una lista di programmi su disco. **Notare che al caricamento della lista un eventuale programma presente nella memoria verrà cancellato.**

Inizializzazione di un disk drive

Quando la spia del disk drive lampeggia, indica che si è verificato un errore di disco. Per riportare il disk drive alla condizione in cui si trovava prima dell'errore, viene usata una procedura chiamata INIZIALIZZAZIONE. Per inizializzare un drive, battere:

OPEN 1,8,15"I":CLOSE 1 RETURN

Se la spia lampeggia ancora, togliere il disco e spegnere e quindi riaccendere il drive.

Per ulteriori informazioni sulle procedure di salvataggio e caricamento dei programmi, consultare il manuale del disk drive o del Datassette e le descrizioni dei comandi SAVE e LOAD nell'Enciclopedia BASIC 7.0 nel Capitolo V.

CAPITOLO 4

USO DELLA MODALITÀ CP/M

SEZIONE 11

Che cos'è il CP/M 3.0

CP/M è un prodotto della Digital Research Inc. La versione del CP/M utilizzata sul Commodore 128 è il CP/M Plus Versione 3.0. In questo capitolo, dove viene trattato per sommi capi l'uso del CP/M sul Commodore 128, il CP/M viene chiamato CP/M 3.0 oppure semplicemente CP/M.

Il CP/M 3.0 è un sistema operativo molto utilizzato nei microcomputer. Come ogni sistema operativo, il CP/M 3.0 gestisce e controlla le risorse del computer, compreso memoria e memorizzazione disco, console (schermo e tastiera), stampante e dispositivi di comunicazione. Il CP/M 3.0 inoltre gestisce le informazioni memorizzate sui file disco. Il CP/M 3.0 può copiare file da un disco alla memoria del computer o ad un dispositivo periferico come ad esempio una stampante. Per eseguire questa operazione il CP/M 3.0 carica diversi programmi in memoria e li esegue in seguito ai comandi introdotti alla console. Dopo il caricamento in memoria il programma esegue una serie di operazioni che ordinano al computer di effettuare un determinato compito.

Il CP/M può essere utilizzato per creare propri programmi oppure si possono utilizzare i moltissimi programmi applicativi CP/M 3.0 già disponibili.

Dispositivi necessari per utilizzare il CP/M 3.0

I dispositivi richiesti per il CP/M 3.0 sono un computer contenente un microprocessore Z80, una console composta da tastiera e schermo ed almeno un disk drive singolo. Per il CP/M 3.0 del Personal Computer Commodore 128 il microprocessore Z80 è incorporato; la console consiste della tastiera completa del Commodore 128 e del monitor a 80 colonne; il disk drive è il nuovo disk drive rapido 1570 della Commodore. Inoltre, nell'imballo del computer sono compresi uno o due dischi CP/M. Se vengono forniti due dischi, uno contiene il sistema CP/M 3.0 e un programma guida (HELP) ed il secondo contiene altri programmi di utilità. Se viene fornito un solo disco, su un lato saranno memorizzati il programma di utilità del sistema ed il programma HELP, mentre sull'altro lato saranno memorizzati gli altri programmi di utilità.

NOTA: Se il CP/M verrà utilizzato con un monitor a 40 colonne, la visualizzazione sarà a 80 colonne, ma verranno visualizzate solo 40 colonne per volta. Per poter visualizzare 80 colonne, si dovrà far scorrere l'immagine orizzontalmente premendo il tasto CONTROL ed il tasto cursore appropriato (verso sinistra o verso destra).

CONTENUTO DEL DISCO

CP/M+.SYS

Si tratta del file di sistema del CP/M Plus principale e contiene tutte le parti del sistema che rimangono permanentemente nella memoria: il Sistema di Input/Output Basic (BIOS) che viene caricato nella parte superiore della memoria, il Sistema Operativo Disco Basic che viene caricato nella memoria immediatamente al di sotto di BIOS, ed i Parametri di Sistema che vengono caricati nell'ultima pagina della memoria.

CCP.COM

Al caricamento del CP/M, il Processore di Comando della Console (CCP) viene caricato nella memoria immediatamente al di sotto di BDOS. La memoria rimanente, sotto il CCP e sopra pagina 0 viene chiamata Area di Programma Transitoria (TPA)ed è il punto in cui vengono caricati i programmi applicativi. CCP è il programma che elabora tutti gli input (generalmente introdotti da tastiera) in risposta al prompt di sistema A>. Il CCP contiene i 6 comandi incorporati (elencati nella Tabella 14.1), ed inoltre supporta i 14 comandi di modifica da console (elencati nella Tabella 13.1).

Il CCP tratta qualsiasi parola (che non sia uno dei comandi incorporati) introdotta in risposta al prompt di sistema come se fosse un comando temporaneo, e quindi tenta la ricerca e l'esecuzione di un file con quel nome e con l'estensione .COM. Se il CCP non trova un file con tale nome sul disco che si sta utilizzando, visualizzerà la parola seguita da un punto interrogativo e quindi dal prompt di sistema.

Se in risposta al prompt di sistema vengono introdotte più parole, tutte le parole dopo la prima verranno trattate come parametri da passare al comando temporaneo.

Un linguaggio o un programma applicativo viene caricato ed eseguito introducendolo come se fosse un comando. Tutti i programmi CP/M includono un file .COM.

CONTENUTO DEL DISCO CP/M 3.0

```
show b:

B: RW, Space       336k

A>dir [full

Scanning Directory...

Sorting Directory...

Directory for Drive A:  User  0

    Name       Bytes   Recs   Attributes      Name       Bytes   Recs   Attributes
------------ ------ ------ ------------ ------------ ------ ------ ------------
CCP      COM     4K     25 Dir RW       CPM+     SYS    23k    182 Dir RW
DIR      COM    15K    114 Dir RW       FORMAT   COM     5k     35 Dir RW
HELP     COM     7k     56 Dir RW       HELP     HLP    83k    664 Dir RW
KEYFIG   COM    10k     75 Dir RW       KEYFIG   HLP     9k     72 Dir RW
PIP      COM     9k     68 Dir RW

Total Bytes     =    165k  Total Records =    1291  Files Found =    9
Total 1k Blocks =    165   Used/Max Dir Entries For Drive A:   15/  64

A>dir b: [full

Scanning Directory...

Sorting Directory...



Directory For Drive B:  User  0

    Name       Bytes   Recs   Attributes      Name       Bytes   Recs   Attributes
------------ ------ ------ ------------ ------------ ------ ------ ------------
DATE     COM     8k     25  Dir RW      DATEC    ASM     2k      5 Dir RW
DATEC    RSX     2k      3  Dir RW      DEVICE   COM    16k     58 Dir RW
DIR      COM    30k    114  Dir RW      DIRLBL   RSX     4k     12 Dir RW
DUMP     COM     2k      8  Dir RW      ED       COM    20k     73 Dir RW
ERASE    COM     8k     29  Dir RW      GENCOM   COM    30k    116 Dir RW
GET      COM    14k     51  Dir RW      INITDIR  COM    64k    250 Dir RW
PATCH    COM     6k     19  Dir RW      PIP      COM    18k     68 Dir RW
PUT      COM    14k     55  Dir RW      RENAME   COM     6k     23 Dir RW
SAVE     COM     4k     14  Dir RW      SET      COM    22k     81 Dir RW
SETDEF   COM     8k     32  Dir RW      SHOW     COM    18k     66 Dir RW
SUBMIT   COM    12k     42  Dir RW      TYPE     COM     6k     24 Dir RW

Total Bytes     =    314k  Total Records =    1168  Files Found =   22
Total 1k Blocks =    314   Used/Max Dir Entries For Drive A:   23/  64
```

File .COM

Tutti gli altri file .COM contengono comandi temporanei (elencati nella Tabella 14.2).

HELP.COM visualizza i messaggi contenuti in HELP.HLP (la cui estensione indica che si tratta di un file dati e non di un file programma), che riguardano il sistema CP/M del C128 ed i suoi comandi. Se non si possiede familiarità con il CP/M e se non si possiedono altri libri o manuali al riguardo, si consiglia di stampare gli schermi di HELP. Per inviare l'output verso stampante, premere CONTROL e P. Premendo nuovamente CONTROL e P si otterrà la disattivazione di tale funzione.

Battere HELP per avere la lista dei diversi argomenti oppure HELP C128 CP/M per informazioni più specifiche (il carattere nel mezzo di C128 CP/M si ottiene premendo la freccia verso sinistra sulla parte superiore della tastiera). In fase di stampa, per non avere pause dopo ogni schermo, introdurre HELP C128 CP/M [NOPAGE.

Per ulteriori dettagli consultare la pagina 14-8.

Altri file

- ASM indica un file sorgente in linguaggio Assembler.
- RSX indica un'estensione di Sistema Residente, cioè un file caricato automaticamente da un file di comando quando richiesto.

Avviamento con il CP/M 3.0

I paragrafi seguenti mostrano come avviare o caricare il CP/M 3.0, come introdurre e modificare una riga di comando e come eseguire copie di riserva dei dischi CP/M3.0.

Caricamento del CP/M 3.0

Caricare il CP/M 3.0 significa leggere una copia del sistema operativo dal disco di sistema CP/M 3.0 alla memoria del computer.

L'operazione di caricamento può essere effettuata in diversi modi. Quando il computer è spento, il CP/M viene caricato accendendo per prima cosa il disk drive ed introducendo il disco di sistema CP/M 3.0, quindi accendendo il computer. Il CP/M 3.0 verrà caricato automaticamente. Se ci si trova già in modalità C128 BASIC, il CP/M 3.0 verrà caricato introducendo il disco di si-

stema CP/M nel drive, battendo il comando BASIC: BOOT seguito da RETURN. A questo punto il CP/M verrà caricato. In modalità C128 il CP/M può essere caricato anche introducendo il disco di sistema e premendo il pulsante RESET.

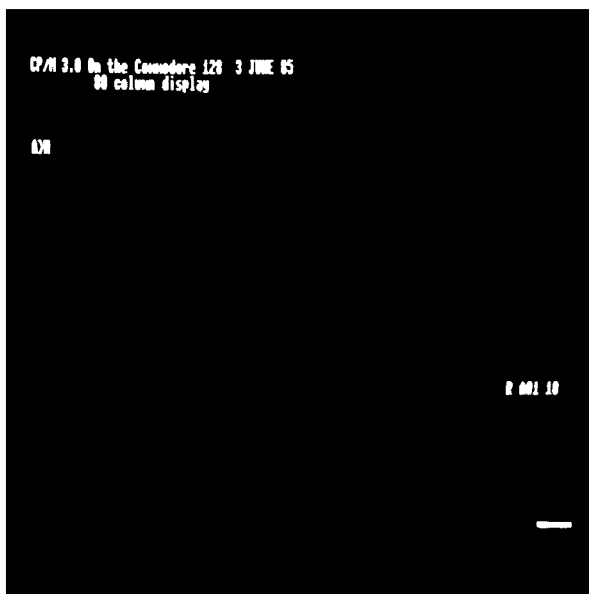
Quando ci si trova in modalità C64 e si desidera attivare la modalità CP/M, per prima cosa occorre spegnere il computer, quindi introdurre il disco di sistema CP/M nel drive ed accendere il computer.

Attenzione: Assicurarsi sempre che il disco sia stato inserito completamente nel drive prima di chiudere lo sportello del drive.

Nel CP/M 3.0 del Commodore 128 l'utente ha a disposizione 59K TPA (area di programma temporanea) che in pratica è la RAM d'utente.

Schermo iniziale del CP/M

Dopo il caricamento del CP/M 3.0 nella memoria, sullo schermo viene visualizzato un messaggio simile a quello che segue:

```
CP/M 3.0 On the Commodore 128  3 JUNE 85
         80 column display
```

Una parte importante del messaggio di avvio è:

A>

Questo è il **prompt di sistema** del CP/M 3.0. Il prompt di sistema informa che il CP/M è pronto a leggere il comando che sarà introdotto da tastiera. Il prompt inoltre informa che il drive A è il drive di default. Questo significa che il CP/M, se non informato diversamente, cercherà i programmi e i file di dati nel drive A. Inoltre il prompt indica che si è stati collegati al sistema come utente 0 in quanto ogni altro numero oltre a 0 non era disponibile.

NOTA: In CP/M un disk drive singolo viene identificato come drive A. A è equivalente al numero di dispositivo 8, drive 0 nelle modalità C128 e C64. Normalmente il numero massimo di drive in CP/M è quattro. Drive addizionali vengono identificati come drive B, C, ecc.

La riga di comando

Il CP/M 3.0 esegue le operazioni a seconda dei comandi specifici introdotti da tastiera. Questi comandi vengono visualizzati sullo schermo in quella che viene chiamata **riga di comando** . Una riga di comando CP/M è composta da una **parola chiave di comando** e da una **coda di comando** opzionale. La parola chiave di comando identifica un comando (programma) che sarà eseguito. La coda di comando può contenere informazioni addizionali per il comando, come ad esempio nomefile o parametri. L'esempio seguente mostra una riga di comando.

A>*DIR MIOFILE*

In questo esempio, DIR è la parola chiave di comando e MIOFILE è la coda di comando. Per inviare la riga di comando al CP/M 3.0 per l'elaborazione, premere il tasto RETURN come indicato dal simbolo RETURN in questo manuale.

Mano a mano che i caratteri vengono battuti da tastiera, saranno visualizzati sullo schermo. Il cursore si muove verso destra. Se si commette un errore di battitura, premere il tasto INST/DEL oppure la combinazione CTRL-H per spostare il cursore verso sinistra e correggere l'errore. CTRL è l'abbreviazione del tasto CONTROL. Per specificare un carattere di controllo, tenere premuto il tasto CTRL e premere il tasto alfabetico appropriato (nella Sezione 13 vengono elencati i caratteri di controllo ed il loro uso).

La parola chiave e la coda di comando possono essere battute in qualsiasi combinazione di lettere maiuscole e minuscole. Il CP/M 3.0 interpreterà tutte le lettere della riga di comando come se fossero maiuscole.

Di solito la riga di comando deve essere battuta subito dopo il prompt di sistema. Comunque, il CP/M 3.0 permette di inserire spazi tra il prompt e la parola chiave di comando.

Tipi di comando

Il CP/M 3.0 riconosce due diversi tipi di comando: i comandi incorporati ed i comandi di utilità temporanei. I comandi **incorporati** eseguono i programmi presenti in memoria come parte del sistema operativo CP/M. Questi comandi possono essere eseguiti istantantaneamente. I comandi **di utilità temporanei** sono memorizzati su disco come file di programma. Per poter eseguire la loro funzione devono essere caricati da disco. I file dei programmi di utilità temporanei sono riconoscibili, dopo la visualizzazione dell'elenco dei file, in quanto i loro nomi sono seguiti da un punto e da COM (.COM). La Sezione 14 fornisce liste dei comandi CP/M incorporati e dei comandi di utilità temporanei.

Per i programmi di utilità temporanei il CP/M 3.0 controlla solamente la parola chiave di comando. Molti programmi di utilità richiedono code di comando esclusive. Se si include una coda di comando, il CP/M 3.0 la passa al programma di utilità senza neppure controllarla. Una coda di comando può contenere un massimo di 128 caratteri.

Come il CP/M legge le righe di comando

Ora verrà utilizzato il comando DIR per mostrare come il CP/M legge le righe di comando. DIR, che è un'abbreviazione di directory (elenco) ordina al CP/M di visualizzare sullo schermo un elenco dei file di disco. Battere la parola chiave DIR dopo la visualizzazione del prompt e premere RETURN:

A>*DIR* **RETURN**

Il CP/M risponde a questo comando visualizzando i nomi di tutti i file che sono memorizzati nel disco che si trova nel drive A. Ad esempio, se nel drive A fosse presente il disco di sistema CP/M, sullo schermo apparirà una lista di nomi di file come quella ripor-

tata di seguito:

A:PIP COM:ED COM:CCP COM:HELP COM:HELP HLP
A:DIR COM:CPM SYS

Il CP/M 3.0 riconosce solo le parole di comando scritte in modo corretto. Se si commette un errore di battitura e si preme RETURN prima di correggerlo, il CP/M 3.0 ripeterà la riga di comando seguita da un punto interrogativo. Ad esempio supponiamo di aver battuto erroneamente il comando DIR, come mostrato di seguito:

A>*DJR* **RETURN**

Il CP/M visualizzerà:

DJR?

Questo significa che il CP/M non può trovare una parola chiave di comando chiamata DJR. Per correggere questo tipo di errori di battitura, utilizzare il tasto INST/DEL per cancellare le lettere errate. Un altro modo per cancellare i caratteri è il tasto CTRL tenuto premuto contemporaneamente al tasto H. Questo sposterà il cursore verso sinistra. Il CP/M fornisce molti altri caratteri di controllo per poter modificare le righe di comando. La Sezione 13 indica come utilizzare i caratteri di controllo per effettuare modifiche sulle righe di comando e sulle informazioni introdotte da console.

DIR accetta come coda di comando un nome di file. DIR può essere utilizzato con un nome di file per controllare se un dato file è presente su disco. Ad esempio, per verificare che il file di programma MIOFILE si trova effettivamente sul disco, battere:

A>*DIR MIOFILE* **RETURN**

Il CP/M 3.0 esegue questa operazione visualizzando il nome del file specificato o il messaggio:

No File

Assicurarsi di aver battuto almeno uno spazio dopo DIR per separare la parola chiave di comando dalla coda di comando. Se non lo si è fatto, il CP/M 3.0 visualizzerà quanto segue:

A>*DIRMIOFILE* **RETURN**
DIRMIOFILE?

COPIA DEL DISCO CP/M 3.0

Prima di eseguire qualsiasi operazione è necessario eseguire una copia del disco di sistema CP/M utilizzando uno o due disk drive. Se si sta utilizzando due drive, questi potrebbero essere due 1541, due 1570 o un 1541 e un 1570. I dischi copia possono essere nuovi o usati. Si possono sia formattare nuovi dischi, sia riformattare dischi già usati. Per eseguire le copie, utilizzare i programmi di utilità FORMAT e PIP sul disco di sistema del CP/M.

Formattazione di un disco

Formattare il dischetto usando il programma FORMAT, con un 1541 o 1570. L'opzione di singola faccia per il C64 è per la formattazione di dischi compatibili con il pacchetto CP/M 2.2 studiato per il Commodore 64.

Introdurre il comando **FORMAT**, selezionare il tipo di disco richiesto utilizzando il tasto di movimento cursore verso il basso, premere RETURN e seguire le istruzioni visualizzate sullo schermo. Premere Y (sì) oppure N (no) in risposta al prompt 'Do you want to format another disk' (si desidera formattare un altro disco?).

Copia di file

Per la copia dei file, utilizzare il Programma PIP, sulla seconda facciata del disco originale. Introdurre PIP ed il consueto prompt (A>) verrà sostituito dal prompt PIP (*).

Se si sta utilizzando un singolo disk drive, usare A come drive sorgente e E come drive destinazione. Il drive E viene chiamato drive virtuale, cioè questo drive non esiste come parte della macchina. Introdurre il disco da copiare nel drive e battere **E:=A:*.*** (verrà visualizzato un messaggio ogni volta che il disco sorgente e il disco destinazione dovranno essere scambiati).

Se si utilizzano due disk drive, introdurre il disco sorgente nel drive A (dispositivo 8) ed il disco appena formattato nel drive B (dispositivo 9 - impostato abbassando l'interruttore DIP sinistro situato sul retro del 1570 mentre il drive è ancora spento) e battere **B:=A:*.*** per copiare tutti i file.

Il disco originale CP/M è un "flippy", cioè è registrato su 2 facce separatamente. Per questo motivo, per leggere la seconda facciata lo si dovrà estrarre dal drive e girare. Questa operazione è

necessaria se si utilizza un 1570 che è un drive a singola faccia. Con questo drive si dovrà quindi copiare le due facce su due dischi separati. Dopo l'operazione di copiatura della prima faccia, girare il disco originale e copiare la seconda faccia sullo stesso disco di destinazione introducendo nuovamente l'istruzione di copia in risposta al prompt del PIP.

Per preparare dischi contenenti solo i file di sistema, utilizzare PIP per copiare i file CPM+.SYS e CCP.COM sul disco appena formattato. NOTA: Questi due file sono necessari solo sui dischi da utilizzare per il caricamento del CP/M.

Utilizzando un drive singolo, battere **E:=A:CPM+.SYS** per copiare il primo file e quindi **E:=A:CCP.COM** per copiare il secondo.

Dopo aver terminato di utilizzare PIP, premere RETURN per tornare al prompt di sistema dal prompt PIP.

Per avere una descrizione completa di PIP, battere HELP PIP, HELP PIP OPTIONS e HELP PIP EXAMPLES.

LINGUAGGI E SOFTWARE APPLICATIVO

Il CP/M è solo un sistema operativo, cioè un mezzo per raggiungere una meta, non la meta stessa. Da solo non può fare nulla. Infatti per scrivere i propri programmi è necessario un linguaggio, assembler o un linguaggio di alto livello. Per divertirsi con i video game o applicare il computer al lavoro, saranno necessari programmi applicativi.

Quali programmi acquistare

Il CP/M è stato implementato su praticamente tutti i computer che utilizzano una cpu Intel 8080 o Zilog Z80. Per questo motivo esiste un grande numero di programmi per l'esecuzione sui sistemi CP/M.Per informazioni rivolgersi presso i migliori rivenditori di programmi.

Il CP/M generalmente utilizza la Modulazione di Frequenza Modificata (MFM) per la registrazione sui dischi. Il DOS Commodore normalmente utilizza la Registrazione Codice Gruppo (GCR). Il disk drive Commodore 1570 può leggere solo la GCR. I pacchetti software CP/M in vendita sono dischi MFM. Anche all'interno della gamma MFM esistono diversi formati: il 1570 può leggere

solo dischi formattati per:

Epson QX10	(settori a 512 byte, doppia faccia, 10 settori per traccia)
IBM 8 SS (CP/M-86)	(settori a 512 byte, singola faccia, 8 settori per traccia)
IBM-8 DS (CP/M-86)	(settori a 512 byte, doppia faccia, 8 settori per traccia)
IBM-9 SS (CP/M-86)	(settori a 512 byte, singola faccia, 9 settori per traccia)
IBM-9 DS (CP/M-86)	(settori a 512 byte, doppia faccia, 9 settori per traccia)
KayPro II	(settori a 512 byte, singola faccia, 10 settori per traccia)
KayPro IV	(settori a 512 byte, doppia faccia, 10 settori per traccia)
Osborne DD SS	(settori a 1024 byte, singola faccia, 5 settori per traccia)
Osborne DD SS	(settori a 1024 byte, doppia faccia, 5 settori per traccia)

Quando si acquista un programma CP/M, lo si deve acquistare su un disco in uno dei formati elencati sopra. Inoltre, ricordare che il C128 eseguirà i programmi scritti per CP/M 2.2 e per CP/M Plus (che è il nuovo nome di quello che originariamente era chiamato versione 3). Comunque, il CP/M-86 è la versione del CP/M progettata per l'utilizzo dei processori a 16 bit: i programmi CP/M-86 non funzioneranno sui processori Z80 a 8 bit del C128, sebbene il 1570 permetterà la lettura dei file dati CP/M-86.

Se si possiede solo il disk drive 1541, si dovrà trasferire i programmi dal formato MFM sul formato GCR. Per questa operazione rivolgersi ai rivenditori di programmi pagando l'eventuale spesa di copiatura.

Installazione sul C128

Esistono moltissimi computer che utilizzano il sistema operativo CP/M. Per questo motivo molti programmi CP/M dovranno essere configurati a seconda dell'apparecchiatura su cui dovranno funzionare. Il processo di installazione di un programma sul C128 implica l'impostazione di parametri all'interno del programma. Il manuale del programma descriverà come installare il programma, se fosse necessaria questa operazione. La maggior parte dei programmi fornisce una lista dei terminali più comuni di supporto.

Se in questa lista fosse incluso anche ADM31, selezionarlo. Se invece non fosse incluso, si dovrà eseguire un'installazione personalizzata.

Di seguito viene fornito un elenco di quanto si dovrà introdurre nel corso dell'esecuzione di WINSTALL.COM (il programma di installazione che fa parte del pacchetto Wordstar), insieme alle informazioni necessarie per l'installazione di altri programmi, sebbene non tutti i programmi formulino le stesse domande.

Terminal name	**Commodore 128**
Screen size	
Screen height	**24**
Screen width	**80**
Cursor positioning	
Function code sequence	**1Bh 3Dh**
Characters to be sent between line number and column number	**none**
Characters to be sent after line number and column number	**none**
Is the column number sent before the line number?	**NO**
What character is sent to the terminal to signify line 1?	**20h**
What character is sent to the terminal to signify column 1?	**20h**
What types of codes are sent to signify line and column numbers?	**Single byte BINARY value**
Terminal start-up	
Function code sequence	**1Bh 59h 1Bh 1Bh 1Bh 60h**
Terminal exit	
Function code sequence	**none**
Highlight-on	
Function code sequence	**1Bh 1Bh 1Bh 52h**
Highlight-off	
Function code sequence	**1Bh 1Bh 1Bh 51h**
Erase to End of Line	**1Bh 54h**
Delete Line	**1Bh 52h**
Insert Line	**1Bh 45h**
Does your terminal use last character on screen as a scroll command?	**YES**

La maggior parte delle stampanti Commodore richiede l'installazione come stampante Standard senza protocollo di comunicazione e dispositivo di lista primaria come driver di Stampante.

NOTA: La lettera h accanto ai numeri nella lista precedente indica che si tratta di numeri esadecimali (che utilizzano cioè base 16 invece della base decimale 10).

SEZIONE 12

Che cos'è un file

Uno dei più importanti compiti del CP/M è di dare accesso e gestire i file su disco. I file in CP/M sono fondamentalmente come i file nelle modalità C128 o C64, cioè una raccolta di informazioni. Comunque, il CP/M gestisce i file in modo diverso in confronto alle modalità C128 e C64. Questa sezione descrive i due tipi di file utilizzati in CP/M; mostra come creare, definire ed accedere a un file; e descrive come i file vengono memorizzati sui dischi CP/M.

Come detto precedentemente, un file CP/M 3.0 è una raccolta di informazioni. Ad ogni file deve essere assegnato un nome esclusivo, attraverso il quale il CP/M potrà identificarlo. Su ogni disco viene memorizzato un elenco. Questo elenco contiene una lista di tutti i nomi di file memorizzati sul disco e le locazioni di ciascun file sul disco stesso.

Esistono due tipi di file CP/M: i file di **programma** (comando) e i file di **dati**. Un file di **programma** contiene una serie di istruzioni che il computer seguirà per raggiungere i risultati desiderati. Un file di **dati** normalmente è una raccolta di informazioni correlate (ad es. una lista di nomi e indirizzi, l'inventario di un magazzino, le registrazioni contabili di un'azienda, il testo di un documento).

Creazione di un file

Esistono diversi modi per creare un file CP/M. Uno è l'utilizzo di un text editor. Il text editor ED del CP/M viene utilizzato per creare e per assegnare un nome ad un file. Un file può anche essere creato copiando in una nuova locazione un file già esistente; in questo processo può essere cambiato anche il nome del file. Con il CP/M, è possibile utilizzare il comando PIP per copiare e ridefinire i file. Infine, alcuni programmi (come MAC, un programma in linguaggio macchina CP/M) creano file di output durante l'elaborazione di file in input.

I comandi ED e PIP vengono trattati brevemente nella Sezione 14 insieme ad altri comandi CP/M utilizzati frequentemente. Per ulteriori dettagli su questi e su tutti gli altri comandi CP/M 3.0 consultare il Manuale per l'Utente del CP/M Plus.

Assegnazione di un nome ad un file

Specifica di file

Il CP/M identifica ogni file per mezzo di un'unica **specifica di file** esclusiva. Una specifica di file può essere composta da quattro parti: l'**identificatore del drive**, il **nome file**, il **tipo file** e la **parola chiave** (password). L'unica parte obbligatoria è il nome di file.

Identificatore del drive

L'identificatore del drive è una lettera singola (dalla A alla P) seguita da un simbolo di due punti. Ad ogni disk drive nel sistema viene assegnata una lettera. Includendo un identificatore del drive come parte della specifica di file, si ordina al CP/M di cercare il file sul disco che si trova al momento nel drive specificato. Ad esempio, battendo:

B: ***MIOFILE*** RETURN

il CP/M cercherà il file MIOFILE nel drive B. Se l'identificatore del drive viene omesso, il CP/M 3.0 cercherà il file nel drive di default (di solito il drive A).

Nome di file

Il nome di file può essere composto da un massimo di otto caratteri, come ad esempio:

MIOFILE

Una specifica di file può consistere semplicemente di un nome di file. Utilizzare come nome di file un nome che ricordi il contenuto del file stesso. Ad esempio, per un file contenente una lista di clienti, si potrà scegliere il nome:

CLIENTI

in modo che il nome indichi il contenuto del file.

Tipo di file

Per poter identificare i file che appartengono ad una stessa categoria, il CP/M permette di aggiungere al nome del file un'estensione composta da uno a tre caratteri chiamata tipo di file. Se si desidera inserire il tipo di file, lo si dovrà battere separandolo dal

nome di file con un punto. Anche in questo caso si consiglia di utilizzare un'estensione che aiuti a riconoscere la categoria del file. Ad esempio, al file contenente la lista dei nomi dei clienti, si potrà aggiungere:

CLIENTI.NOM

Quando il CP/M visualizza le specifiche di file, aggiungerà degli spazi ai nomi di file corti, in modo da poter confrontare con facilità i tipi di file. I file di **programma** che il CP/M carica nella memoria dal disco hanno il tipo di file COM. Non utilizzare questo tipo di file nelle proprie specifiche.

Parola chiave

Nel CP/M 3.0 del Commodore 128 è possibile includere una parola chiave come parte della specifica di file. La parola chiave può essere composta da un massimo di otto caratteri e dovrà essere separata dal tipo di file (o dal nome di file se non viene specificato alcun tipo di file) per mezzo di un punto e virgola, come mostrato di seguito:

CLIENTI.NOM;CONTABIL

La parola chiave è opzionale, comunque se un file è stato protetto con una parola chiave, per poter accedere a questo file si DEVE introdurre la parola chiave come parte della specifica di file.

Esempio di una specifica di file

Una specifica di file contenente tutti i quattro possibili elementi consiste di un identificativo del drive, di un nome di file primario, di un tipo di file e di una parola chiave, tutti separati dai caratteri o dai simboli appropriati come mostrato nell'esempio seguente:

A:DOCUMENT.LEG;MARIO RETURN

Numero di utente

Il CP/M 3.0 identifica tutti i file anche assegnando a ciascuno un numero di utente compreso tra 0 e 15. Il CP/M 3.0 assegna il numero di utente ad un file nel corso della creazione del file stesso. I numeri di utente permettono di separare i file in 16 gruppi.

Il numero di utente precede sempre l'identificativo del drive fatta eccezione per l'utente 0, che è il numero di utente di default e non viene visualizzato nel prompt. Di seguito vengono forniti alcuni esempi di numeri di utente e del loro significato.

4A> Numero di utente 4, drive A
A> Numero di utente 0, drive A
2B> Numero di utente 2, drive B

Per cambiare il numero corrente di utente, utilizzare il comando incorporato USER, come segue:

A> ***USER 3*** **RETURN**
3A>

Il numero di utente ed il drive possono essere modificati introducendo il nuovo numero e l'identificativo del drive quando viene visualizzato il prompt di sistema:

A>***3B:*** **RETURN**
3B>

Con la maggior parte dei comandi è possibile avere accesso solo ai file con il numero di utente corrente. Comunque, se un file risiede in utente 0 ed è contrassegnato da un attributo di file, sarà possibile accedere a questo file da qualsiasi numero di utente.

Uso di caratteri variabili per avere accesso a più di un file

Alcuni comandi incorporati e temporanei CP/M 3.0 possono selezionare ed elaborare diversi file se nel nome o nel tipo di file vengono inclusi caratteri variabili speciali. Un carattere variabile è un carattere che può essere utilizzato al posto di altri caratteri. Il CP/M 3.0 utilizza l'asterisco (*) ed il punto interrogativo (?) come caratteri variabili. Ad esempio, utilizzando un ? come terzo carattere di un nome di file, si comunica al CP/M che ? sta per **qualsiasi** carattere che potrà essere incontrato in quella posizione. Allo stesso modo un * ordina al CP/M di riempire il nome di file con un numero di punti interrogativi sufficiente a coprire l'intero nome di file. Una specifica di file contenente caratteri variabili viene chiamata specifica di file ambigua e può riferirsi a più file in quanto fornisce al CP/M 3.0 un modello di corrispondenza per trovare il nome di file. Il CP/M 3.0 ricerca all'interno dell'elenco del disco e seleziona tutti i file il cui nome o tipo corrisponda al modello fornito. Ad esempio, battendo:

?????TAS.LIB **RETURN**

il CP/M 3.0 selezionerà tutti i file il cui nome termina con TAS e il cui tipo sia .LIB.

Caratteri riservati

I caratteri mostrati nella Tabella 12-1 hanno un significato particolare per il CP/M 3.0. Per questo motivo questi caratteri non dovranno essere utilizzati nelle specifiche di file tranne dove indicato.

Tabella 12-1. Caratteri riservati CP/M 3.0

Carattere	Significato
<$,![>[] tabulazione spazio ritorno a capo	delimitatori di specifica di file
:	delimitatore di drive nella specifica di file
.	delimitatore di tipo di file nella specifica di file
;	delimitatore di parola chiave nella specifica di file
;	delimitatore di commento all'inizio di una riga di comando
*?	caratteri variabili in una specifica di file ambigua
<>&!]+–	delimitatori della lista di opzioni
[]	delimitatori della lista di opzioni per opzioni globali e parziali
()	delimitatori per i modificatori multipli all'interno di parentesi quadre per opzioni con modificatori
/$	delimitatori di opzioni in una riga di comando.

Tipi di file riservati

Il CP/M 3.0 ha già stabilito diversi gruppi di file. La Tabella 12-1 elenca alcuni dei tipi di file con una breve descrizione per ognuno di essi.

Tabella 12-2. Tipi di file riservati CP/M 3.0

Tipo di file	Significato
ASM	File sorgente Assembler
BAS	Programma sorgente BASIC
COM	Programma di Z80 o di linguaggio macchina equivalente
HEX	File di output da MAC (utilizzato da HEXCOM)
HLP	File di messaggio HELP
$$$	File temporaneo
PRN	File di stampa MAC o RMAC
REL	File di output da RMAL (utilizzato da LINK)
SUB	Lista di comandi che devono essere eseguiti da SUBMIT
SYM	File simbolo da MAC, RMAC, o LINK
SYS	File di sistema

SEZIONE 13

Questa sezione descrive come il CP/M 3.0 comunica con la console e la stampante, come avviare e arrestare l'output di console e stampante e come modificare i comandi introdotti da tastiera.

Controllo dell'output su console

A volte le informazioni visualizzate dal CP/M 3.0 scompaiono dallo schermo troppo velocemente e quindi non possono essere lette. Per ordinare al sistema di attendere prima di terminare la visualizzazione, premere contemporaneamente i tasti CTRL e S. La sequenza CTRL-S arresta la visualizzazione. Quando si desidera continuare, premere CTRL-Q. Premendo il tasto NO SCROLL si otterrà una pausa del sistema e la visualizzazione di una finestra di "**pausa**" sulla riga di stato in fondo allo schermo (riga 25). Per riprendere la visualizzazione, premere nuovamente NO SCROLL. Se verrà premuto un qualsiasi tasto diverso da CTRL-Q o NO SCROLL durante una pausa di visualizzazione, il CP/M 3.0 emetterà un segnale acustico.

Alcuni programmi di utilità CP/M 3.0 (come ad esempio DIR e TYPE) hanno un sistema di impaginazione automatica alla console. Questo significa che se l'output del programma è più lungo di quanto possa essere contenuto in uno schermo, la visualizzazione si arresterà automaticamente quando lo schermo sarà riempito completamente. In questo caso per riprendere la visualizzazione si dovrà premere RETURN. Questa opzione può essere attivata o disattivata per mezzo del comando SETDEF.

Controllo dell'output su stampante

Per ottenere l'**eco** su stampante è possibile utilizzare anche un comando di controllo. Per avviare l'eco su stampante, premere CTRL-P. Per informare l'utente che l'eco è stata attivata, il sistema emette un segnale acustico. Per arrestare questa funzione, premere nuovamente CTRL-P (in questa fase non viene emesso alcun segnale acustico). Mentre l'eco su stampante è attiva, tutti i caratteri che vengono visualizzati sullo schermo saranno stampati.

L'eco su stampante può essere utilizzata con un comando DIR per ottenere una lista dei file memorizzati su un floppy disk. Per ottenere una copia su carta di una parte di file, si potranno utilizzare anche CTRL-P con CTRL-S e CTRL-Q. Per avviare la visualizzazione del file su schermo, usare il comando TYPE. Quando viene raggiunto il punto che si desidera stampare premere CTRL-S per

interrompere la visualizzazione, CTRL-P per avviare l'eco su stampante e quindi CTRL-Q per riprendere la visualizzazione ed iniziare la stampa. Per disattivare l'eco su stampante si potrà utilizzare un'altra sequenza CTRL-S, CTRL-P, CTRL-Q.

Modifica di una riga da console

Come si è visto precedentemente, è possibile correggere gli errori di battitura utilizzando il tasto INST DEL o CTRL-H. Il CP/M 3.0 possiede funzioni di correzione riga addizionali che vengono eseguite per mezzo di caratteri di controllo. I caratteri di controllo possono essere usati per modificare righe di comando o righe di input nella maggior parte dei programmi.

Uso dei caratteri di controllo per la modifica di una riga

Per mezzo dei caratteri di controllo per la modifica di una riga elencati nella Tabella 13-1 si potrà muovere il cursore verso sinistra e verso destra per inserire e cancellare caratteri all'interno di una riga di comando. In questo modo si eviterà di dover ribattere tutto quello che si trova alla destra della correzione effettuata.

Nell'esempio seguente l'utente scrive PIP in modo errato e il CP/M 3.0 visualizza un messaggio di errore. L'utente richiama la riga di comando contenente l'errore premendo CTRL-W e corregge l'errore (il carattere di sottolineatura rappresenta il cursore):

A>POP A:=B:*.* _	(errore di battitura in PIP)
POP?	
A>POP A:=B:*.* _	(CTRL-W richiama la riga)
A>POP A:=B:*.*	(CTRL-B sposta il cursore all'inizio della riga)
A>POP A:=B:*.*	(CTRL-F sposta il cursore verso destra)
A>PP A:=B:*.*	(CTRL-G cancella l'errore)
A>PIP A:=B:*.*	(battendo "I" il nome di comando viene corretto)

Dopo la correzione della riga di comando, l'utente può premere RETURN anche se il cursore si trova al centro della riga. Battendo RETURN (o uno dei caratteri di controllo equivalenti) non solo si ottiene l'esecuzione del comando, ma si ottiene anche la memorizzazione del comando in un buffer in modo che sia possibile premere CTRL-W per un'eventuale correzione o riesecuzione.

Quando si inserisce un carattere all'interno di una riga, i caratteri che si trovano a destra del cursore si spostano verso destra. Se la riga dovesse diventare troppo lunga per la larghezza dello schermo, i caratteri scompariranno nel lato destro dello schermo. Questi caratteri però non vengono persi, infatti ricompariranno cancellando alcuni caratteri dalla riga o premendo CTRL-E quando il cursore si trova al centro della riga. CTRL-E provoca lo spostamento dei caratteri a destra del cursore sulla riga sottostante.

La Tabella 13-1 fornisce una lista completa dei caratteri di controllo CP/M 3.0 per la modifica di una riga con il Commodore 128.

Tabella 13-1. Caratteri di controllo per la modifica di una riga con il CP/M 3.0

Carattere	**Significato**
CTRL-A	Sposta il cursore di un carattere verso sinistra.
CTRL-B	Sposta il cursore all'inizio della riga di comando senza alcun effetto sul contenuto della riga stessa. Se il cursore si trova all'inizio della riga, CTRL-B lo sposterà alla fine della riga.
CTRL-E	Provoca un ritorno a capo forzato ma non invia la riga di comando al CP/M 3.0. Sposta il cursore all'inizio della riga sottostante senza cancellare l'input precedente.
CTRL-F	Sposta il cursore di un carattere verso destra.
CTRL-G	Cancella il carattere su cui è posizionato il cursore. Il cursore non si sposta. I caratteri alla destra del cursore si spostano di uno spazio verso sinistra.
CTRL-H	Cancella il carattere alla sinistra del cursore e sposta il cursore a sinistra di uno spazio. I caratteri a destra del cursore si spostano di uno spazio verso sinistra.
CTRL-I	Sposta il cursore al prossimo punto di tabulazione. I punti di tabulazione vengono impostati automaticamente ogni otto colonne. Questa combinazione ha lo stesso effetto del tasto TAB.

CTRL-J	Invia la riga di comando al CP/M 3.0 e posiziona il cursore all'inizio della riga sottostante. Questa combinazione ha lo stesso effetto di un RETURN o di CTRL-M.
CTRL-K	Cancella tutta la riga a partire dalla posizione del cursore.
CTRL-M	Invia la riga di comando a CP/M 3.0 e riporta il cursore all'inizio della riga sottostante. Questa combinazione ha lo stesso effetto di un RETURN o di CTRL-J.
CTRL-R	Batte nuovamente la riga di comando. Inserisce un carattere # alla posizione in cui si trova il cursore, sposta il cursore sulla riga sottostante e ribatte ogni comando parziale battuto fino a quel momento.
CTRL-U	Cancella tutti i caratteri nella riga di comando, inserisce un carattere # alla posizione corrente del cursore e sposta il cursore sulla riga sottostante. È comunque possibile utilizzare la combinazione CTRL-W per richiamare qualsiasi carattere che si trova a sinistra del cursore quando è stato premuto CTRL-U.
CTRL-W	Richiama e visualizza la riga di comando precedentemente introdotta sia al livello del sistema operativo, sia nel corso dell'esecuzione di un programma, se CTRL-W è il primo carattere introdotto dopo il prompt. CTRL-J, CTRL-M, CTRL-U e RETURN definiscono la riga di comando che si può richiamare. Se la riga di comando contiene alcuni caratteri, CTRL-W sposta il cursore alla fine della riga di comando. Premendo RETURN, CP/M 3.0 eseguirà il comando richiamato.
CTRL-X	Cancella tutti i caratteri a sinistra del cursore e sposta il cursore all'inizio della riga corrente. CTRL-X salva i caratteri a destra del cursore.

SEZIONE 14

Come si è visto nella Sezione 11, una riga di comando CP/M 3.0 consiste di una parola chiave di comando, di una coda di comando opzionale e da una pressione del tasto RETURN. Questa sezione descrive i due tipi di comandi che la parola chiave di comando può identificare, e fornisce alcune informazioni generali sui comandi individuali e sulle loro funzioni. La sezione inoltre fornisce alcuni esempi dei comandi più utilizzati e spiega il concetto di dispositivi logici e fisici del CP/M 3.0. Questa sezione indica anche come il CP/M 3.0 ricerca un file di programma sul disco, come eseguire comandi multipli e come ripristinare il sistema operativo. Infine, la sezione spiega come utilizzare il comando HELP direttamente da tastiera per avere informazioni sui vari argomenti riguardanti il CP/M, compreso i formati di comando ed il loro utilizzo.

I due tipi di comandi del CP/M 3.0

Il CP/M 3.0 comprende due tipi di comandi:

- **I comandi incorporati** – che identificano i programmi in memoria.
- **I comandi di utilità temporanei** – che identificano i file di programma sul disco.

Il CP/M 3.0 possiede sei comandi incorporati e più di 20 comandi di utilità temporanei. Al sistema possono essere aggiunti i programmi di utilità acquistando i diversi programmi applicativi compatibili con il CP/M 3.0. Gli utenti programmatori potranno scrivere i propri programmi di utilità da utilizzare con il CP/M 3.0.

Comandi incorporati

I comandi incorporati sono componenti del CP/M 3.0 che possono essere utilizzati sempre, indipendentemente dal disco presente nel disk drive. I comandi incorporati vengono introdotti nella memoria del computer al caricamento del CP/M 3.0 e vengono eseguiti più velocemente dei programmi di utilità temporanei. La Tabella 14-1 elenca i comandi incorporati del CP/M 3.0 Commodore 128.

Alcuni comandi incorporati possiedono opzioni che richiedono supporto da parte di un programma di utilità temporaneo. Il comando di utilità temporaneo relativo ha lo stesso nome del comando incorporato ed ha il tipo di file COM.

Tabella 14-1. Comandi incorporati

Comando	Funzione
DIR	Visualizza i nomi di file di tutti i file presenti nell'elenco, tranne quelli contrassegnati dall'attributo SYS.
DIRSYS	Visualizza i nomi di file dei file contrassegnati dall'attributo (di sistema) SYS nell'elenco.
ERASE	Cancella un nome di file dall'elenco del disco liberando lo spazio di memoria occupato dal file.
RENAME	Assegna un nuovo nome a un file disco.
TYPE	Visualizza sullo schermo il contenuto di un file ASCII (TESTO).
USER	Si sposta su un diverso numero di utente.

Comandi di utilità temporanei

I comandi di utilità temporanei del CP/M 3.0 sono elencati nella Tabella 14-2. Quando viene introdotta una parola chiave di comando che identifica un'utilità temporanea, il CP/M 3.0 carica il file di programma dal disco e passa su questo file il nome di file, i dati o i parametri che sono stati introdotti nella coda di comando. DIR, RENAME e TYPE sono comandi incorporati con estensioni transitorie opzionali.

Tabella 14-2. Comandi di utilità temporanei

Comando	Funzione
DATE	Imposta o visualizza la data e l'ora.
DEVICE	Assegna un dispositivo logico CP/M ad uno o più dispositivi fisici, modifica il protocollo del driver e le velocità di trasmissione, o imposta l'ampiezza di schermo.
DIR	Visualizza l'elenco dei file e le loro caratteristiche.
DUMP	Visualizza un file in formato ASCII ed esadecimale.

ED	Crea e modifica i file ASCII.
ERASE	Utilizzato per la cancellazione di caratteri variabili.
FORMAT	Formatta un disco CP/M. Cancella i dati dai dischi usati.
GENCOM	Crea un file speciale COM con un file RSX accodato.
GET	Riceve temporaneamente un input da un file disco piuttosto che da tastiera.
HELP	Visualizza le informazioni su come utilizzare i comandi CP/M 3.0.
INITDIR	Inizializza un elenco di disco per permettere di stampare l'ora e la data.
KEY FIG	Permette la modifica della definizione dei tasti.
PATCH	Visualizza o installa routine di correzione al sistema CP/M.
PIP	Copia e combina i file.
PUT	Invia temporaneamente l'output della stampante o della console ad un file disco.
RENAME	Modifica il nome di un file o di un gruppo di file per mezzo di caratteri variabili.
SAVE	Copia il contenuto della memoria in un file.
SET	Imposta le opzioni di file compreso etichette dei dischi, attributi di file, forma in cui viene visualizzata la data e l'ora e protezione della parola chiave.
SETDEF	Imposta le opzioni di sistema compreso la catena di ricerca del drive.
SHOW	Visualizza i dati di disco e drive.
SUBMIT	Esegue automaticamente comandi multipli.
TYPE	Visualizza sullo schermo (o stampa, se richiesto) il contenuto di file di testo (o di gruppi di file, se vengono utilizzati caratteri variabili).

Commutazione dell'input e dell'output

Il comando PUT del CP/M 3.0 permette di dirigere l'output su console o stampante su un file di disco. È possibile utilizzare un comando GET per fare prelevare al CP/M 3.0 o ad un programma di utilità l'input da un file disco. Gli esempi che seguono mostrano alcune delle possibilità di GET e PUT.

Il comando PUT può essere utilizzato per inviare un ouput su console ad un file di disco o alla console. Con il comando PUT si potrà creare un file di disco contenente l'elenco di tutti i file su quel disco, come mostrato nella Figura 14-1.

A>PUT CONSOLE OUTPUT TO FILE DIR.PRN
INVIO DELL'OUTPUT SU CONSOLE AL FILE: DIR.PRN

```
A>DIR
A:FILENAME   TEX:FRONT   TEX:FRONT    BAK:ONE      BAK:THREE   TEX
A:FOUR       TEX:ONE     TEX:LINEDIT  TEX:EXAMP1   TXT:TWO     BAK
A:TWO        TEX:THREE   BAK:EXAMP2   TXT

A>TYPE DIR.PRN
A:FILENAME   TEX:FRONT   TEX:FRONT    BAK:ONE      BAK:THREE   BAK
A:FOUR       TEX:ONE     TEX:LINEDIT  TEX:EXAMP1   TXT:TWO     BAK
A:TWO        TEX:THREE   BAK:EXAMP2   TXT
```

Figura 14-1. Esempio del comando PUT

Un comando GET può ordinare al CP/M 3.0 o ad un programma di leggere l'input della console da un file di disco invece che da tastiera. Se il file dovrà essere letto dal CP/M 3.0, dovrà contenere le righe di comando standard del CP/M 3.0. Se il file dovrà essere letto da un programma di utilità, dovrà contenere l'input appropriato per quel programma. Un file può contenere sia righe di comando CP/M 3.0 sia un input di programma se però contiene anche un comando di avviamento del programma.

Assegnazione dei dispositivi logici

La configurazione hardware minima per il CP/M 3.0 con il Commodore 128 comprende una console costituita da tastiera e schermo, e un disk drive 1570. Al sistema si potrà aggiungere un altro dispositivo, come ad esempio una stampante o un modem. La Tabella 14-3 fornisce i nomi dei dispositivi logici CP/M 3.0 con i diversi dispositivi fisici di input e output. Inoltre la tabella mostra i dispositivi fisici assegnati ai dispositivi logici del sistema CP/M 3.0 del Commodore 128.

Tabella 14-3. Dispositivi logici CP/M 3.0

Nome del dispositivo logico	Tipo di dispositivo	Assegnazione del dispositivo fisico
CONIN:	Input da console	Tastiera
CONOUT:	Output su console	Schermo a 80 colonne
AUXIN:	Input ausiliario	Nessuna
AUXOUT:	Output ausiliario	Nessuna
LST:	Output lista	PRT1 oppure PRT2

Queste assegnazioni possono essere modificate tramite un comando DEVICE. Ad esempio, è possibile assegnare AUXIN e AUXOUT ad un modem in modo che il computer possa utilizzare le linee telefoniche per comunicare con altri utenti di computer o con servizi informativi.

Ricerca dei file di programma

Se una parola chiave di comando identifica un programma di utilità, il CP/M 3.0 ricercherà il file sul drive di default o sul drive di utente corrente e quindi sotto l'utente 0 per lo stesso file contrassegnato dall'attributo SYS. In qualsiasi momento durante il processo di ricerca, il CP/M 3.0 arresta la ricerca al ritrovamento del file di programma. Quindi il CP/M 3.0 carica il programma in memoria e lo esegue. Al termine del programma, il CP/M 3.0 visualizza il prompt di sistema ed attende il prossimo comando. Comunque, se il CP/M 3.0 non trova la riga di comando, ripeterà la riga di comando seguita da un punto interrogativo ed attenderà il prossimo comando.

Esecuzione di comandi multipli

Negli esempi visti fino a questo momento, il CP/M 3.0 ha eseguito solo un comando alla volta. Il CP/M 3.0 però può eseguire anche sequenze di comandi. La sequenza di comandi viene introdotta in risposta al prompt di sistema oppure è possibile memorizzare su un file disco una sequenza di comandi utilizzata frequentemente specificando "SUB" come tipo di file. Dopo aver memorizzato la sequenza su un file disco, questa potrà essere eseguita in ogni momento per mezzo di un comando SUBMIT.

Chiusura del programma

Per terminare l'esecuzione di un programma o ripristinare il sistema operativo viene utilizzata la combinazione di tasti CTRL-C introdotta tenendo premuto il tasto CTRL contemporaneamente a C.

La maggior parte dei programmi applicativi che funzionano con il CP/M e molti programmi di utilità temporanei possono essere conclusi con la combinazione CTRL-C. Se però si tenta di concludere un programma mentre è in corso la visualizzazione su schermo, si dovrà premere CTRL-S per arrestare la visualizzazione prima di introdurre CTRL-C.

Come ottenere aiuto

Il CP/M 3.0 possiede un comando di utilità transitorio chiamato HELP che visualizza un riassunto del formato e dell'utilizzo dei più usati comandi CP/M. Per accedere a HELP, battere il seguente comando:

A>*HELP* RETURN

È possibile premere il tasto HELP invece di battere da tastiera la parola HELP seguita da RETURN.

Viene così visualizzata la lista dei comandi, come mostrato di seguito:

Argomenti disponibili:

COMMANDS	CNTRLCHARS	DATE	DEVICE	DIR	
DUMP	ED	ERASE	FILESPEC	GENCOM	GET
HELP	HEXCOM	INITDIR	LIB	LINK	MAC
PATCH	PIP (COPY)	PUT	RENAME	RMAC	SAVE
SET	SETDEF	SHOW	SID	SUBMIT	TYPE
USER	XREF				

Ad esempio, battendo:

HELP>PIP RETURN

il CP/M visualizzerà le seguenti informazioni:

PIP (COPY)

Syntax:

DESTINATION SOURCE

PIP d: Gn filespec [Gn] = filespec [o],...d:[o]

Spiegazione:

Il programma di copia file, PIP, copia i file, li combina e trasferisce i file da dischi, stampanti, console o altri dispositivi collegati al computer. La prima specifica di file (filespec) è la destinazione. La seconda è la sorgente. Battere due o più filespec sorgenti separate da virgole per combinare uno o più file in un altro file. [o] è una qualsiasi combinazione delle opzioni disponibili. L'opzione [Gn] nella specifica di file di destinazione ordina a PIP di copiare il file su quel numero di utente.

PIP senza coda di comando visualizza un * ed attende la serie di comandi, introdotti ed elaborati uno alla volta. La sorgente o la destinazione possono essere un dispositivo logico qualsiasi del CP/M 3.0.

La funzione HELP fornisce informazioni come quelle riportate nell'esempio su tutti i comandi incorporati ed i comandi di utilità termporanei del CP/M 3.0. Se si desiderano le informazioni su un'area specifica, battere **HELP argomento** dopo il prompt di sistema, dove "argomento" è una coda di comando che indica l'argo-

mento su cui si desiderano informazioni. Ad esempio:

A>*HELP PIP*
A>*HELP DIRSYS*

HELP può essere utilizzato ogni volta che si desiderano informazioni su un comando specifico, oppure può essere utilizzato per ampliare le proprie conoscenze del CP/M 3.0.

SEZIONE 15

Tastiera

La Commdore ha fornito il CP/M di diversi ampliamenti. Questi miglioramenti combinano le caratteristiche del Commodore 128 a quelle del CP/M 3.0. Questa sezione descrive gli ampliamenti apportati.

Tutti i tasti della tastiera possono essere definiti per poter produrre un codice o una funzione, **tranne** i tasti seguenti:

Tasto SHIFT sinistro
Tasto SHIFT destro
Tasto Commodore
Tasto CTRL
Tasto RESTORE
Tasto 40/80
Tasto CAPS LOCK

Nella definizione di un tasto, la tastiera riconosce le seguenti funzioni speciali. Per specificare queste funzioni, tenere premuto il tasto CTRL e il tasto SHIFT destro, contemporaneamente al tasto della funzione desiderata.

Tasto	Funzione
Tasto CURSORE A SINISTRA	**Definisce il tasto**
Tasto CURSORE A DESTRA	**Definisce la stringa**
Tasto ALT	**Alterna le funzioni dei tasti**

Definizione di un tasto

L'utente può definire il codice prodotto da un tasto. Ogni tasto possiede quattro definizioni possibili: Normale, Maiuscolo Alfabetico con Shift e con Control. Il Maiuscolo Alfabetico con Shift viene attivato/disattivato premendo il tasto Commodore. Dopo l'attivazione di questa modalità, nella parte inferiore dello schermo apparirà un riquadro. Il primo tasto premuto sarà il tasto da definire. Viene visualizzato il valore esadecimale (HEX) corrente per il tasto; a questo punto l'utente può battere il nuovo codice HEX per il tasto, oppure annullare l'operazione premendo un tasto non esadecimale. Di seguito viene fornita una definizione dei codici che possono essere assegnati ad un tasto (in modalità ALT i codici vengono rinviati all'applicazione; vedere la Modalità ALT più avanti).

Codice	Funzione
00h	Nullo (uguale a non premere alcun tasto)
da 01h a 7Fh	Codici ASCII normali
da 80h a 9Fh	Assegnazione di stringa
da A0h a AFh	Colore carattere (80 colonne)
da B0h a BFh	Colore sfondo (80 colonne)
da C0h a CFh	Colore carattere (40 colonne)
da D0h a DFh	Colore sfondo (40 colonne)
da E0h a EFh	Colore cornice (40 colonne)
F0h	Attivazione/disattivazione stato disco
F1h	Pausa di sistema
F2h	(Non definito)
F3h	Finestra di schermo destra 40 colonne
F4h	Finestra di schermo sinistra 40 colonne
da F5h a FFh	(Non definito)

Definizione di una stringa

Questa funzione permette di assegnare più di un codice tasto ad un singolo tasto. Tutti i tasti premuti in questa modalità vengono introdotti nella stringa. I caratteri battuti vengono visualizzati in un riquadro nella parte inferiore dello schermo.

NOTA: Alcuni tasti potrebbero non visualizzare il simbolo riportato su ciascuno di essi. Per fornire all'utente il controllo sul processo di introduzione dati, sono disponibili cinque tasti funzione speciali. Per poter accedere a queste funzioni, premere il tasto CTRL ed il tasto SHIFT destro insieme ai tasti funzione desiderati.

Tasto	Funzione
RETURN	Completa la definizione di stringa
+ (tastiera principale)	Inserisce dello spazio nella stringa
– (tastiera principale)	Cancella il carattere cursore
Freccia a sinistra	Cursore a sinistra
Freccia a destra	Cursore a destra

Uso della modalità ALT

La modalità ALT è una funzione di commutazione. Il valore di default è OFF. Questa funzione permette di inviare codici a 8 bit a un'applicazione.

Schermo

Lo schermo di default del CP/M 3.0 emula un terminale ADM31. Le seguenti funzioni di schermo emulano il funzionamento con ADM 3A, che è un sottoinsieme delle operazioni su terminale ADM31.

CTRLG	Segnale acustico
CTRL H	Cursore a sinistra
CTRL J	Cursore verso il basso
CTRL K	Cursore verso l'alto
CTRL L	Cursore a destra
CTRL M	Cursore all'inizio della riga corrente (CR)
CTRL Z	Cursore in posizione HOME e cancellazione dello schermo
ESC = RC	Posizione del cursore, dove R è la locazione di riga (con valori da spazio a 8) e C è la locazione di colonna (valori da spazio a 0) in relazione alla riga di stato

Altre funzioni in modalità ADM31 sono:

ESC T	
ESC t	Cancellazione fino alla fine della riga
ESC Y	
ESC y	Cancellazione fino alla fine dello schermo
ESC:	Cursore nella posizione Home e cancellazione
ESC*	dello schermo (compresa la riga di stato)
ESC Q	Inserimento caratteri
ESC W	Cancellazione caratteri
ESC E	Inserimento riga
ESC R	Cancellazione riga

* ESC ESC ESC color# imposta uno dei 16 colori di schermo visti nel Capitolo II, Sezione 6, Figura 6-2. Il numero del colore sarà impostato come segue:

da 20h a 2Fh	colore carattere
da 30h a 3Fh	colore sfondo
da 40h a 4Fh	colore cornice (solo per modalità 40 colonne)

L'effetto visivo prodotto dalle funzioni seguenti può essere ottenuto solo con il formato di schermo a 80 colonne.

ESC>	Mezza intensità
ESC<	Piena intensità
ESC G4	Attivazione inversione video
* ESC G3	Attivazione sottolineatura
ESC G2	Attivazione lampeggiamento
* ESC G1	Selezione del set di caratteri alternativo
ESC G0	Disattivazione di tutti gli attributi ESC G.

* **NOTA:** Queste non sono normali sequenze ADM31.

****

Le sezioni di questo capitolo forniscono informazioni generali sulla struttura e delle vaste possibilità del CP/M 3.0.

CAPITOLO 5

ENCICLOPEDIA BASIC 7.0

SEZIONE 16

Struttura dell'Enciclopedia

Questo capitolo illustra gli elementi del linguaggio BASIC 7.0, fornendo le regole di sintassi del BASIC 7.0 per il Commodore 128 ed una descrizione sommaria per ognuno di questi elementi.

Il BASIC 7.0 comprende tutti gli elementi del BASIC 2.0.

Gli elementi principali del BASIC sono elencati nelle seguenti sezioni distinte:

1. **COMANDI e ISTRUZIONI:** comandi utilizzati per creare, memorizzare e cancellare programmi e istruzioni di programma BASIC utilizzate in righe di programma numerate.
2. **FUNZIONI:** funzioni di stringa, di stampa e funzioni numeriche.
3. **VARIABILI e OPERATORI:** tutti i vari tipi di variabili, nomi ammessi di variabili, operatori aritmetici e logici.
4. **PAROLE e SIMBOLI RISERVATI:** parole e simboli riservati del linguaggio BASIC 7.0 che possono essere utilizzati per altri scopi.

Formato dei comandi e delle istruzioni

Comando →	**AUTO**
Descrizione →	– Attiva/disattiva la numerazione automatica di riga
Formato →	**AUTO[riga#]**
Spiegazione →	Questo comando attiva la funzione di numerazione automatica di riga. Questo facilita l'operazione di introduzione dei programmi inserendo automaticamente i numeri di riga al posto dell'utente ogni volta che viene premuto il tasto RETURN alla fine di ogni riga di programma. Il cursore si sposterà a destra del numero di due spazi. L'argomento "riga#" si riferisce all'incremento desiderato tra i numeri di riga. AUTO senza l'argomento "riga#" disattiva la numerazione automatica che può anche essere disattivata da RUN. Questa istruzione può essere utilizzata solo in modalità diretta (all'esterno di un programma).
Esempi →	**ESEMPI:** **AUTO 10:** Numera automaticamente le righe di programma con incrementi di 10. **AUTO 50:** Numera automaticamente le righe di programma con incrementi di 50. **AUTO :** Disattiva la numerazione automatica di riga.

La riga in neretto che definisce il formato consiste dei seguenti elementi:

Le parti del comando o dell'istruzione, che devono essere battute esattamente come mostrato, sono scritte a lettere maiuscole. Le parole fornite dall'utente, come ad esempio il nome del programma, sono minuscole.

Quando viene indicato qualcosa tra virgolette (" ") (di solito un nome di programma o di file) è necessario riportarte le virgolette come mostrato nell'esempio di formato.

Le **PAROLE CHIAVE**, chiamate anche parole riservate, appaiono a lettere maiuscole. Le parole chiave possono essere battute sia utilizzando la parola intera, sia l'abbreviazione prevista (per una lista completa delle abbreviazioni consultare l'Appendice K). La parola chiave o l'abbreviazione deve essere introdotta correttamente, altrimenti verrà visualizzato un messaggio di errore. I messaggi di errore BASIC e DOS vengono definiti rispettivamente nelle Appendici A e B.

Le parole chiave sono parte integrante del linguaggio BASIC. Sono la parte centrale del comando o dell'istruzione, ed indicano al computer l'azione da intraprendere. Queste parole non possono essere utilizzate come nomi di variabili. Per una lista completa delle parole e dei simboli riservati consultare la Sezione 20.

Gli **ARGOMENTI**, chiamati anche parametri, vengono indicati a lettere maiuscole. Gli argomenti sono complementari delle parole chiave fornendo informazioni specifiche al comando o all'istruzione. Ad esempio, la parola chiave "load" indica al computer di caricare un programma e l'argomento indica il nome specifico del programma. Un secondo argomento specifica da quale drive caricare il programma. Gli argomenti comprendono nomi di file, variabili, numeri di riga, ecc.

Le **PARENTESI QUADRE []** indicano gli argomenti opzionali. L'utente seleziona un argomento oppure nessun argomento, a seconda delle necessità.

Le **PARENTESI ANGOLARI** <> indicano che l'utente DEVE scegliere uno degli argomenti elencati.

Una **BARRA VERTICALE** | separa gli elementi in una lista di argomenti quando le scelte sono limitate agli argomenti elencati,. Quando la barra verticale appare in una lista racchiusa tra PARENTESI QUADRE, le scelte saranno limitate agli elementi della lista, ma l'utente avrà ancora la possibilità di non utilizzare alcun argomento.

I **PUNTI DI SOSPENSIONE ...** indicano che un'opzione o un argomento può essere ripetuto più di una volta.

Le **VIRGOLETTE " "**racchiudono stringhe di caratteri, nomi di file ed altre espressioni. Quando gli argomenti sono racchiusi tra virgolette, queste devono essere indicate nel comando o nell'istruzione. Le virgolette non sono convenzioni utilizzate per descrivere i formati, ma bensì parti integranti del comando o dell'istruzione.

Le **PARENTESI ()**, quando racchiudono argomenti, devono essere indicate nel comando o nell'istruzione. Le parentesi non sono convenzioni utilizzate per descrivere i formati, ma bensì parti integranti di un comando.

VARIABILE si riferisce a qualsiasi nome valido di variabile BASIC, come ad esempio X,A$,T%,ecc.

ESPRESSIONE si riferisce a qualsiasi espressione valida BASIC come ad esempio A+B+2,.5*(X+3), ecc.

VIRGOLE (,), DUE PUNTI (:) e **PUNTI E VIRGOLA (;)** devono essere inclusi nei comandi o istruzioni essendo parte integrante di comandi o istruzioni.

Formato del comando grafico e sonoro

I parametri opzionali nei comandi grafici e sonori sono rappresentati come segue:

[,parametro]

Quando i parametri vengono omessi, si dovrà introdurre la virgola, questo perché i parametri sono legati a una specifica posizione. Non si dovrà però includere la virgola dopo l'ultimo parametro specificato.

ESEMPIO

ENVELOPE n [,att] [,dec] [,sos] [,ril] [,fo] [,li]

Per modificare solo il parametro ril, utilizzare:

ENVELOPE n,,,,ril

Le prime tre virgole indicano la posizione di attacco, decadimento, sostegno e la terza indica la posizione di rilascio. Non devono essere introdotte le virgole per forma d'onda (fo) e larghezza dell'impulso (li).

Nei comandi grafici, ogni volta che viene specificata una coordinata (per mezzo di X e Y), è possibile sostituire questa coordinata con un vettore (X;Y); in questo caso X è la distanza (scalata) e Y è l'angolo in gradi (0=su 90=destra, ecc.).

ESEMPIO:

LOCATE 160,100
DRAW TO 40;45

traccerà una riga a 45 gradi con lunghezza 40.

Formato del comando disco

I parametri opzionali nei comandi disco vengono indicati come segue:

[,parametro]

La virgola non è obbligatoria se il parametro è il primo subito dopo il comando. Se vengono omessi altri parametri che richiedono la virgola, sarà omessa anche la virgola stessa.

ESEMPIO:

DIRECTORY [Ddrive][<ON|,>Un numero dispositivo] [,carattere variabile]

produrrà:

DIRECTORY DO ON U8,"AB*"

Per specificare solo il carattere variabile, la virgola non è richiesta.

ESEMPIO:

DIRECTORY "AB*"

Quando vengono utilizzate variabili all'interno di comandi disco, queste dovranno essere racchiuse tra parentesi (   ).

ESEMPIO:

DIRECTORY D(DV),(A$)

SEZIONE 17

APPEND

APPEND #numero di file logico,"nomefile"[,Dnumero drive] [<ON|,>Udispositivo]

Questo comando apre il file definito da nomefile e posiziona il puntatore alla fine del file stesso. Successive istruzioni di scrittura PRINT# provocheranno l'accodamento alla fine del file indicato. I valori di default per il numero di drive e di dispositivo sono rispettivamente 0 e 8.

Le variabili o le espressioni utilizzate come nomi di file devono essere racchiuse tra parentesi.

ESEMPI:	
Append # 8, "MIOFILE"	Apre il file logico 8 definito "MIOFILE" per l'accodamento di successive istruzioni PRINT#.
Append # 7, (A$),D0,U9	Apre il file logico definito dalla variabile in A$ sul drive 0, numero di dispositivo 9 e prepara l'accodamento.

AUTO

– Attiva/disattiva la numerazione automatica di riga

AUTO[riga#]

Questo comando attiva la funzione di numerazione automatica di riga. Questo facilita l'operazione di introduzione dei programmi inserendo automaticamente i numeri di riga al posto dell'utente ogni volta che viene premuto il tasto RETURN alla fine di ogni riga di programma. Il cursore si sposterà sul secondo spazio a destra del numero. L'argomento "riga#" si riferisce all'incremento desiderato tra i numeri di riga. AUTO senza l'argomento "riga#" disattiva la numerazione automatica che può anche essere disattivata da RUN. Questa istruzione può essere utilizzata solo in modalità diretta (all'esterno di un programma).

ESEMPI:

AUTO 10	Numera automaticamente le righe di programma con incrementi di 10.

AUTO 50	Numera automaticamente le righe di programma con incrementi di 50.
AUTO	Disattiva la numerazione automatica di riga.

BACKUP

– Copia il contenuto di un disco su un altro su un disk drive doppio

BACKUP sorgente Dnumero drive TO destinazione Dnumero drive [<ON|,>Udispositivo]

Questo comando copia tutti i dati dal dischetto sorgente al dischetto destinazione utilizzando un disk drive doppio. Con il comando BACKUP può essere utilizzato un dischetto nuovo senza prima doverlo formattare. Questo perchè il comando BACKUP copia tutte le informazioni contenute nel dischetto compreso il formato. Per questo motivo il comando BACKUP distrugge tutte le informazioni eventualmente presenti sul dischetto destinazione. Quindi utilizzando questo comando con un dischetto usato, assicurarsi che questo non contenga programmi che si desiderano conservare. Per precauzioni il computer chiederà conferma ("ARE YOU SURE?") prima di iniziare l'operazione. Premere il tasto "Y" per eseguire il comando, o qualsiasi altro tasto per annullarlo. È consigliabile creare sempre una copia dei propri dischi in caso il dischetto originale venga perso o danneggiato. Vedere anche il comando COPY. Il numero di dispositivo di default è 8.

NOTA: Questo comando può essere utilizzato solo con un disk drive doppio. Questo comando non potrà essere utilizzato per l'esecuzione di copie di dischi protetti (come la maggior parte dei programmi in commercio).

ESEMPI:

BACKUP D0 to D1	Copia tutti i dati dal disco nel drive 0 al disco nel drive 1 nel disk drive doppio 8.
BACKUP D0 TO D1 ON U9	Copia tutti i dati dal drive 0 al drive 1 nel disk drive 9.

BANK

– Seleziona uno dei 16 banchi (da 0 a 15)

BANK numero banco

Questa istruzione specifica il numero di banco e la configurazione di memoria corrispondente per la memoria del Commodore 128. Il banco di default è 15. Qui di seguito viene fornita una tabella delle configurazioni BANK disponibili nella memoria del Commodore 128:

BANCO	CONFIGURAZIONE
0	Solo RAM(0)
1	Solo RAM(1)
2	Solo RAM(2)*
3	Solo RAM(3)*
4	ROM interna, RAM(0), I/O
5	ROM interna, RAM(1), I/O
6	ROM interna, RAM(2), I/O*
7	ROM interna, RAM(3), I/O*
8	ROM esterna, RAM(0), I/O
9	ROM esterna, RAM(1), I/O
10	ROM esterna, RAM(2), I/O*
11	ROM esterna, RAM(3), I/O*
12	ROM interna e Kernal (BASSA), RAM(0), I/O
13	ROM esterna e Kernal (BASSA), RAM(0), I/O
14	ROM BASIC e Kernal, RAM(0), ROM carattere
15	ROM BASIC e Kernal, RAM(0), I/O

* Per utilizzo su versioni con memoria interna più estesa, es. 256K. Nelle versioni non ampliate, non esiste RAM in questi banchi e 2 dà come eco 0 e 3 dà come eco 1.

Per accedere ad un banco particolare battere BANK n (n=0–15), quindi utilizzare PEEK/POKE o SYS. Dal monitor far precedere una cifra esadecimale (0–F) al numero esadecimale a quattro cifre della gamma di indirizzo che si sta visualizzando.

BEGIN/BEND

Struttura utilizzata con IF...THEN ELSE in modo da poter includere diverse righe di programma tra l'inizio (BEGIN) e la fine (BEND) della struttura.

IF Condizione THEN BEGIN:istruzione

istruzione
istruzione BEND:ELSE BEGIN
istruzione
istruzione BEND

ESEMPIO:

```
10 IF X=1 THEN BEGIN:PRINT "X=1 è vera"
20 PRINT "Viene eseguita questa parte dell'istruzione"
30 PRINT "Quando X è uguale a 1"
40 BEND:PRINT "Fine della struttura BEGIN/BEND":GOTO
   60
50 PRINT "X non è uguale a 1":PRINT "Le istruzioni tra BEGIN
   e BEND vengono ignorate"
60 PRINT "Parte rimanente del programma"
```

Se l'istruzione condizionale (IF...THEN) nella riga 10 è vera, vengono eseguite le istruzioni tra le parole chiave BEGIN e BEND, comprese tutte le istruzioni sulla stessa riga di BEND. Se l'istruzione condizionale (IF...THEN) nella riga 10 è falsa, vengono ignorate tutte le istruzioni tra BEGIN e BEND, compreso quelle sulla stessa riga di programma di BEND ed il programma riprende dalla prima riga di programma immediatamente successiva a quella contenente di BEND. BEGIN/BEND tratta le righe da 10 a 40 come riga unica.

Le stesse regole vengono applicate se viene specificata una clausola ELSE:BEGIN. Se la condizione è vera, vengono eseguite tutte le istruzioni tra ELSE:BEGIN e BEND comprese le istruzioni sulla stessa riga di BEND. Se è falsa, il programma riprende dalla riga immediatamente successiva a quella contenente BEND.

BLOAD

– Carica un file binario iniziando alla locazione di memoria specificata.

BLOAD "nomefile"[,Dnumero drive][<ON|,>Unumero dispositivo][,Bnumero banco][,Pindirizzo iniziale]

dove:

- nomefile è il nome di un file
- numero banco permette di selezionare uno dei 16 banchi
- indirizzo iniziale è la locazione di memoria dove inizia il caricamento

Un file binario è un file (di programma o di dati) che è stato salvato per mezzo del monitor di linguaggio macchina o per mezzo del comando BSAVE. Il comando BLOAD carica il file binario alla locazione specificata dall'indirizzo iniziale.

ESEMPI:

BLOAD "SPRITE", B0, P3584	Carica il file finario "SPRITE" iniziando dalla locazione 3584 nel banco zero.
BLOAD "DATI1", D0, U8, B1, P4096	Carica il file binario "DATI1" nella locazione 4096 (BANCO 1) dal drive 0, unità 8.

Non specificando l'indirizzo iniziale, il file comunicherà allo stesso indirizzo da cui era stato salvato.

BOOT

– Carica ed esegue un programma salvato come file binario

BOOT["nomefile"][,Dnumero drive][<ON|,>Udispositivo]

Il comando carica un file binario eseguibile e avvia l'esecuzione all'indirizzo iniziale predefinito. Il numero di dispositivo di default è 8, drive 0.

ESEMPI:

BOOT	Carica e lancia un programma eseguibile (ad e. CP/M Plus). Questo è un caso speciale e richiede l'impostazione di un settore specifico sul disco.
BOOT "GRAPHICS1", D0 U9	Carica e lancia il programma "GRAPHICS1" dall'unità 9, drive 0.

L'esecuzione inizia all'indirizzo iniziale della riga di programma da cui inizia il caricamento.

BOX

– Traccia un riquadro alla posizione specificata sullo schermo

BOX[sorgente colore],X1,Y1[,X2,Y2][,angolo][,colorazione]

dove:

sorgente colore	0 = Colore sfondo 1 = Colore primo piano 2 = Multicolor 1 } solo nella modalità 3 = Multicolor 2 } grafiche 3 e 4
x1,y1	Coordinata dell'angolo superiore sinistro (scalata)
x2,y2	Angolo inferiore destro opposto a x1, y1 (scalato), il default è la locazione CP.
angolo	Rotazione dei gradi in senso orario; il default è 0 gradi.
colorazione	Colorazione dell'immagine 0 = Non colorare 1 = Colorare (default = 0)

Questa istruzione permette all'utente di tracciare sullo schermo un rettangolo di una qualsiasi dimensione. La rotazione è basata sul centro del rettangolo. Il cursore pixel (CP) è posizionato a x2,y2 dopo l'esecuzione dell'istruzione BOX. Il numero della sorgente di colore deve essere zero (0) o uno (1) se in modalità a matrice di punti, oppure un 2 o un 3 se in modalità a matrice di punti multicolore. Vedere anche il comando GRAPHIC per la selezione della modalità grafica appropriata da utilizzare con il numero della sorgente di colore BOX.

Vedere anche il comando LOCATE per ulteriori informazioni sul cursore pixel:

ESEMPI:

BOX 1,10,10,60,60	Traccia il perimetro di un rettangolo.
BOX 1, 10,10,60,60,45,1	Traccia un riquadro colorato ruotato (un rombo).
BOX, 30,90,,45,1	Traccia un poligono pieno, ruotato.
BOX 1,20,20,,,1	Traccia un rettangolo pieno a 20 pixel a destra e a 20 pixel verso il basso rispetto alla posizione corrente del cursore pixel.

Qualsiasi parametro può essere omesso a patto che al suo posto venga introdotta una virgola, come negli ultimi due esempi.

N.B.: x2,y2 vengono trattati come un solo parametro, per questo è necessaria una sola virgola. Si verificherà un cambiamento di riga se il valore del grado è maggiore di 360, cioè 360=0 (450=90).

BSAVE

– Salva un file binario dalle locazioni di memoria specificate.

BSAVE"nomefile"[,Dnumero drive][,<ON|,>Unumero dispositivo][,Bnumero banco],Pindirizzo iniziale TO Pindirizzo finale

dove:

- nomefile è il nome assegnato al file
- numero drive è 1 o 0 per un disk drive doppio (0 è il default per un disk drive singolo)
- dispositivo numero è il numero del disk drive (il default è 8)
- numero banco è il numero di banco specificato (0-15)
- indirizzo iniziale è l'indirizzo di partenza da cui viene salvato il programma
- indirizzo finale è l'ultimo indirizzo salvato in memoria, cioè l'indirizzo finale deve essere superiore di un byte rispetto all'ultimo byte da salvare.

Questo è quanto avviene utilizzando il comando SAVE nel MONITOR del linguaggio macchina.

ESEMPI:

BSAVE "DATI SPRITE",B0,P3584 TO P4096

Salva il file binario chiamato "DATI SPRITE", a partire dalla locazione 3584 fino alla 4095 (BANCO 0).

BSAVE"PROGRAMMA.SCR",D0,U9,B0,P3182 TO P8000

Salva il file binario chiamato "PROGRAMMA.SCR" nella gamma dell'indirizzo di memoria da 3182 a 7999 (BANCO 0) sul drive 0, unità 9.

CATALOG

– Visualizza l'elenco del disco

CATALOG[Dnumero drive][<ON|,>Unumero dispositivo] [,stringa variabile]

Il comando CATALOG visualizza l'elenco sul drive specificato, esattamente come il comando DIRECTORY. Vedere il comando DIRECTORY per avere altri esempi (DIRECTORY e CATALOG sono intercambiabili).

ESEMPIO:

CATALOG	Visualizza l'elenco del disco sul drive 0, unità 8.

CHAR

– Visualizza sullo schermo i caratteri alla posizione specificata

CHAR[sorgente colore][,x,y[,stringa][,RVS]

Questa funzione è stata progettata principalmente per la visualizzazione dei caratteri su uno schermo a matrice di punti, ma può essere utilizzata anche su uno schermo di testo. Di seguito viene fornito il significato dei parametri:

sorgente colore	0 = Sfondo 1 = Primo piano 2 = Multicolor 1 3 = Multicolor 2
x	Colonna carattere (0-79) (in modalità a 40 colonne salta alla riga sottostante)
y	Riga carattere (0-24)
stringa	Stringa da stampare
inversione	Flag campo invertito (0=disattivato, 1=attivato)

Il testo (stringhe alfanumeriche) può essere visualizzato su qualsiasi tipo di schermo in qualsiasi locazione per mezzo dell'istruzione CHAR. I dati di carattere sono letti dall'area ROM di carattere del Commodore 128. L'utente introduce le coordinate x e y del punto di partenza e indica la stringa di testo da visualizzare. La sorgente di colore e l'inversine dell'immagine sono opzionali.

La stringa viene continuata sulla riga sottostante se supera il margine destro dello schermo. Quando viene utilizzata in modalità TESTO, la stringa stampata dal comando CHAR funziona esattamente come una stringa PRINT, con le stesse funzioni di cursore e controllo colore. Queste funzioni di controllo all'interno della stringa non sono attive quando il comando CHAR viene utilizzato per visualizzare il testo in modalità a matrice di punti. Anche i controlli maiuscolo/minuscolo (CHR$(14) o CHR$(142)) funzionano in modalità a matrice di punti.

I caratteri multicolore vengono gestiti diversamente dai caratteri standard. La tabella riportata di seguito mostra come ottenere le combinazioni possibili.

			Flag di inversione 0 (OFF)	Flag di inversione 1 (ON)
Sorgente colore	0	Testo	1	1
		Fondo	2	3
Sorgente colore	1	Tesio	1	0
		Fondo	0	1
Sorgente colore	2	Testo	2	0
		Fondo	0	2
Sorgente colore	3	Testo	3	0
		Fondo	0	3

ESEMPIO:

```
10 COLOR 2,3:REM multicolor 1=Rosso
20 COLOR 3,7:REM multicolor 2=Blu
30 GRAPHIC 3,1
40 CHAR 0,10,10,"TESTO",0
50 CHAR 0,10,11,"TESTO",1
```

CIRCLE

– Traccia sullo schermo cerchi, ellissi, archi, ecc. alle posizioni specificate.

CIRCLE[sorgente colore],X,Y,Xr[,Yr][,ap][,a][,angolo][,inc]

dove:

sorgente colore .... 0 = colore sfondo
1 = colore primo piano
2 = multicolor 1 } Modalità
3 = multicolor 2 } grafiche 3 e 4

x,y .... Coordinata centrale del cerchio

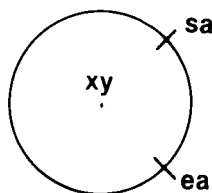
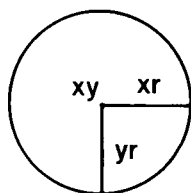
xr .... Raggio X (scalato)

yr .... Raggio Y (default = xr)

ap .... Angolo di partenza arco (default = 0 gradi)

aa .... Angolo di arrivo arco (default 360 gradi)

angolo .... Rotazione in senso orario in gradi (default = 0 gradi)

inc .... Gradi tra segmenti (default = 2 gradi)

Con l'istruzione CIRCLE l'utente può tracciare un cerchio, un'ellisse, un arco, un triangolo, un ottagono o altri poligoni. Il cursore pixel (CP) si trova alla circonferenza del cerchio sull'angolo di arrivo dell'arco. Tutte le rotazioni sono relative al centro. Gli archi vengono tracciati dall'angolo di partenza a quello di arrivo in senso orario. L'incremento controlla la regolarità della forma; utilizzando valori più bassi si otterranno delle figure più precise. Specificando per inc un valore maggiore di 2 si otterrà una figura con una forma più approssimativa.

Vedere anche il comando LOCATE per ulteriori dettagli sul cursore pixel.

ESEMPI:

CIRCLE1, 160,100,65,10	Traccia un'ellisse.
CIRCLE1, 160,100,65	Traccia un cerchio.
CIRCLE1, 60,40,20,18,,,,45	Traccia un ottagono.
CIRCLE1, 260,40,20,30,,,,90	Traccia un rombo.
CIRCLE1, 60,140,20,18,,,,120	Traccia un triangolo.

È possibile omettere un parametro, ma in questo caso al suo posto si dovrà introdurre una virgola. I parametri omessi prenderanno i valori di default.

CLOSE

– Chiude un file logico

CLOSE numero file

Questa istruzione chiude i file utilizzati dalle istruzioni DOPEN o OPEN. Il numero che segue la parola CLOSE è il numero del file da chiudere.

ESEMPIO:

CLOSE 2	Il file logico 2 viene chiuso.

CLR

– Cancella le variabili di programma

CLR

Questa istruzione cancella tutte le variabili presenti in memoria, lasciando il programma intatto. CLR viene eseguita automaticamente quando viene emesso un comando RUN o NEW.

CMD

– Ridirige l'output di schermo

CMD numero file logico [,lista di scrittura]

Questo comando invia l'output, che generalmente è diretto allo schermo (ad es. l'istruzione PRINT produrrà sempre un listato, ma non sullo schermo) ad un altro dispositivo, come ad esempio un file dati disco o una stampante. Questo dispositivo o file dovrà però essere prima aperto. Il comando CMD deve essere seguito da un numero o da una variabile numerica che faccia riferimento al file. La lista di scrittura può essere una variabile o una stringa alfanumerica. Questo comando è utile per la stampa di intestazioni di listati di programma.

ESEMPIO:

OPEN,1,4	Apre il dispositivo #4 (stampante).
CMD 1	Tutto l'output normale va ora alla stampante.
LIST	Il listato va alla stampante e non allo schermo (anche la parola READY).
PRINT#1	Invia l'output nuovamente allo schermo.
CLOSE 1	Chiude il file.

COLLECT

– Libera lo spazio inaccessibile su disco

COLLECT[Dnumero drive][<ON|,>Udispositivo]

Utilizzare questo comando per liberare lo spazio su disco che è stato allocato ai file chiusi impropriamente e per cancellare i riferimenti a questi file nell'elenco. Questo tipo di file viene evidenziato nell'elenco da un asterisco dopo il nome. Il dispositivo di default è il numero 8.

ESEMPIO:

COLLECT D0	Libera tutto lo spazio disponibile che è stato allocato per i file chiusi impropriamente.

N.B.: Questo comando libererà anche lo spazio allocato per l'accesso diretto. Per ulteriori dettagli consultare il manuale del disk drive.

COLLISION

– Definisce la gestione degli interrupt di collisione sprite

COLLISION tipo[,istruzione]

tipo	Tipo di interrupt, come segue: 1 = Collisione sprite/sprite 2 = Collisione sprite/dati di visualizzazione 3 = Penna ottica
istruzione	Numero di riga BASIC di una subroutine.

Quando si verifica la situazione specificata, il BASIC terminerà l'elaborazione dell'istruzione in corso di esecuzione ed effettuerà un GOSUB al numero di riga specificato. Al termine della subroutine (che dovrà finire con un RETURN) il BASIC riprenderà l'elaborazione dal punto di interruzione. L'azione di interrupt continuerà finché non verrà specificata un'istruzione COLLISION dello stesso tipo senza numero di riga. Possono essere attivati più tipi di

interrupt contemporaneamente, ma ognuno sarà gestito individualmente (cioè non sarà possibile alcuna ricursione o nidificazione degli interrupt). La causa dell'interrupt potrà continuare a provocare altri interrupt per un certo periodo di tempo a meno che la situazione non venga modificata oppure gli interrupt non siano disattivati.

Per determinare quale sprite è stato coinvolto in una collisione da quando è stato effettuato l'ultimo controllo, utilizzare la funzione BUMP.

ESEMPIO:

Collision 1,5000	Rileva una collisione sprite/sprite ed il controllo del programma viene inviato alla subroutine nella riga 5000.
Collision 1	Arresta l'interrupt provocato dall'esempio precedente.
Collision 2,1000	Rileva una collisione sprite/dati ed il controllo del programma viene inviato alla subroutine nella riga 1000.

NOTA: Gli sprite possono entrare in collisione anche quando si trovano al di fuori del campo visivo, a meno che non siano stati disattivati.

COLOR

– Definisce i colori per ogni area dello schermo

COLOR numero sorgente, numero colore

Questa istruzione assegna un colore ad una delle sette aree di colore:

Area	Sorgente
0	Sfondo - 40 colonne (VIC)
1	Primo piano - 40 colonne (VIC)
2	Multicolor 1
3	Multicolor 2
4	Cornice - 40 colonne (VIC)
5	Colore carattere (schermo a 40 o 80 colonne)
6	Colore di sfondo - 80 colonne

I colori utilizzabili sono compresi nella gamma da 1 a 16

Codice Colore	Colore	Codice Colore	Colore
1	Nero	9	Arancio
2	Bianco	10	Marrone
3	Rosso	11	Rosso chiaro
4	Azzurro	12	Grigio scuro
5	Porpora	13	Grigio
6	Verde	14	Verde chiaro
7	Blu	15	Blu chiaro
8	Giallo	16	Grigio chiaro

Numeri dei colori nel formato a 40 colonne

Codice Colore	Colore	Codice Colore	Colore
1	Nero	9	Porpora scuro
2	Bianco	10	Marrone
3	Rosso scuro	11	Rosso chiaro
4	Azzurro chiaro	12	Azzurro scuro
5	Porpora chiaro	13	Grigio
6	Verde chiaro	14	Verde chiaro
7	Blu scuro	15	Blu chiaro
8	Giallo chiaro	16	Grigio chiaro

Numeri dei colori nel formato a 80 colonne

ESEMPIO:

Color 0,1: Modifica il colore dello sfondo in nero (40 colonne).

Color 5,8: Modifica il colore del carattere in giallo.

CONCAT

– Concatena due file dati

CONCAT"file 2"[,Dnumero drive]TO"file 1"[,Dnumero drive] [<ON|,>Udispositivo]

Il comando CONCAT collega il file 2 alla fine del file 1 mantenendo il nome del file 1. Il numero di dispositivo di default è 8 ed il numero del drive di default è 0.

ESEMPIO:

Concat "File B" to "File A"	Il FILE B viene collegato al FILE A ed il file risultante viene chiamato FILE A.
Concat (A$) to (B$), D1, U9	Il file denominato B$ diventa un nuovo file con lo stesso nome del file denominato da A$ collegato alla fine di B$ - Questo viene eseguito sull'Unità 9, drive 1 (disk drive doppio).

Quando viene utilizzata una variabile come nome di file, come nell'ultimo esempio, la variabile del nome di file deve essere racchiusa tra parentesi.

NOTA: Utilizzare nomi di file corti (10 caratteri max) in quanto il buffer di comando in alcuni disk drive è limitato.

CONT

– Continua l'esecuzione del programma

CONT

Questo comando viene utilizzato per riavviare un programma che era stato interrotto da un tasto STOP, da un'istruzione STOP oppure da un'istruzione END. Il programma riprende l'esecuzione dal punto in cui era stato interrotto. CONT non riprenderà l'esecuzione se sono state modificate delle righe, se alcuni righe sono state aggiunte al programma oppure se sullo schermo è stata effettuata qualche operazione di modifica del programma. Se il programma si era arrestato a causa di un errore; oppure se si è commesso un errore prima di tentare il riavvio del programma, CONT non avrà alcun effetto. In questo caso il messaggio di errore visualizzato sarà: CAN'T CONTINUE ERROR.

COPY

– Copia i file da un drive ad un altro su disk drive doppi o singoli.

COPY <["nomefile sorgente"][,Dnumero drive]>TO< ["nomefile destinazione"][,Dnumero drive][<ON|,> Udispositivo]

Questo comando copia i file da un disco (file sorgente) ad un altro (file destinazione) utilizzando un disk drive doppio. Con un drive singolo è possibile creare una copia del file sullo stesso disco, utilizzando però un altro nome di file. Il nome di file potrà però essere lo stesso se si effettuerà l'operazione di copiatura da un drive ad un altro.

Il comando COPY può anche copiare tutti i file da un drive all'altro con un disk drive doppio. In questo caso i numeri di drive vengono specificati ed i nomi di file di sorgente e destinazione vengono omessi.

I parametri di default per il comando COPY sono: 8 per il numero di dispositivo e 0 per il drive.

NOTA: La copia tra due unità a singolo o doppio drive non è possibile. Vedere BACKUP.

ESEMPI:

COPY"prova",D0 TO "prog prova",D1	Copia "prova" dal drive 0 al drive 1, assegnando il nome "prog prova" al file sul drive 1.
COPY"programma",D0 TO "programma",D1	Copia "programma" dal drive 0 al drive 1.
COPY D0 TO D1	Copia tutti i file dal drive 0 al drive 1.
COPY"prog.lav." TO "backup"	Copia "prog.lav." come file chiamato "backup" sullo stesso disco (drive 0).

DATA

– Definisce i dati da utilizzare da parte del programma

DATA lista delle costanti

Questa istruzione è seguita da una lista di elementi di dati da introdurre nella memoria del computer dalle istruzioni READ. Gli elementi possono essere numerici o di stringa e vengono separati da virgole. I dati di stringa non devono necessariamente essere racchiusi tra virgolette, a meno che contengano uno dei caratteri seguenti: spazio, due punti, virgola. Se non viene introdotto alcun dato tra due virgole, il valore verrà letto come zero se si tratta di un valore numerico, oppure come stringa vuota. Vedere anche l'istruzione RESTORE, che permette al Commodore 128 di rileggere i dati.

ESEMPIO:

DATA 100,200,MARIO,"CIAO,MAMMA",,3,14,ABC123

DCLEAR

– Svuota tutti i canali aperti sul disk drive

DCLEAR [Dnumero drive][<ON|,>Udispositivo]

Questa istruzione chiude e svuota tutti i canali aperti sul numero di dispositivo specificato. Il default è U8. Questo comando è analogo a OPEN 0,8,15,"I0" CLOSE 10.

ESEMPI:

DCLEAR D0	Cancella tutti i canali aperti sul drive 0, numero di dispositivo 8.
DCLEAR D1,U9	Cancella tutti file canali aperti sul drive 1, numero di dispositivo 9.

NOTA: Tutti i file saranno cancellati; i dati non potranno essere recuperati dai file su cui erano stati scritti inizialmente. Vedere CLOSE/DCLOSE.

DCLOSE

– Chiude il file disco

DCLOSE[#numero file logico][<ON|,>,Udispositivo]

Questa istruzione chiude un file singolo o tutti i file correntemente aperti su un'unità a disco. Se non viene specificato alcun numero di file logico, verranno chiusi tutti i file aperti correntemente. Il numero di dispositivo di default è 8. Notare gli esempi seguenti:

ESEMPI:

DCLOSE	Chiude tutti i file aperti correntemente sull'unità 8.
DCLOSE #2	Chiude i file associati al numero file logico 2.
DCLOSE ON U9	Chiude tutti i file correntemente aperti sull'unità 9.

DEF FN

– Ritorna il valore di una funzione definita dall'utente

DEF FN nome (variabile) = espressione

Questa istruzione permette la definizione di un calcolo complesso come funzione. Nel caso di una formula lunga utilizzata più volte all'interno di uno stesso programma, questa parola chiave permette di risparmiare spazio di programma. Il nome assegnato alla funzione inizia con le lettere FN, seguite da un qualsiasi nome di variabile numerica permesso. Per prima cosa definire la funzione utilizzando l'istruzione DEF seguita dal nome assegnato alla funzione. Dopo il nome viene introdotta una coppia di parentesi () che racchiudono un nome di variabile numerica fittizio. A questa variabile sarà assegnato un valore solo se appare nell'espressione a destra del simbolo di uguale (=). Lo stesso nome di variabile può essere utilizzato in altri punti del programma ma sarà separato completamente dal suo uso nella funzione. Nelle espressioni possono essere utilizzate altre variabili e/o funzioni, che vengono valutate al richiamo della funzione. Quindi occorre introdurre un simbolo di uguale, seguito dalla formula da definire. La funzione può essere eseguita sostituendo a X un qualsiasi numero, utilizzando il formato mostrato nelle righe 20, 40 e 50 del programma mostrato di seguito.

ESEMPIO:

10 DEF FNEG(LO)=INT((V1∗ LO 2)∗100)/100	Lo è un valore locale per questa linea
20 LO=15	Variabile di programma normale
30 V1=3.14159	P greco approssimativo
40 PRINT FNEG(5)	Assegna nella funzione il valore 5 al valore locale L0
50 PRINT FNEG (1)	Utilizza 1 invece di L0 nella funzione
60 PRINT INT((V1∗ LO 2)∗100/100	Variabile L0 utilizzata
70 PRINT	Rimane invariato

DELETE

– Cancella le righe di un programma BASIC nella gamma specificata.

DELETE<[riga iniziale][riga iniziale-][riga iniziale-riga finale]| [-riga finale]>

Questo comando può essere eseguito solo in modalità diretta.

ESEMPI:

DELETE 75	Cancella la riga 75.
DELETE 10-50	Cancella le righe da 10 a 50 compresa.
DELETE-50	Cancella tutte le righe dall'inizio del programma fino alla riga 50 compresa.
DELETE 75-	Cancella tutte le righe dalla 75 all'ultima riga del programma compresa.

DIM

– Dichiara il numero degli elementi in una matrice

DIM variabile (indici) [,variabile(indici)][...]

Prima di poter utilizzare le matrici di variabili, il programma dovrà eseguire un'istruzione DIM per stabilire le dimensioni delle matrici (a meno che in ogni dimensione della matrice stessa non siano presenti 11 o meno di 11 elementi). L'istruzione DIM è seguita dal nome della matrice, che può essere un qualsiasi nome di variabile permesso. Quindi, racchiuso tra parentesi, si dovrà introdurre il numero (o variabile numerica) degli elementi in ogni dimensione. Una matrice con più di una dimensione viene chiamata matrice multidimensionale. Può essere utilizzato un numero qualsiasi di dimensioni, tenendo però presente che l'intera lista di variabili create occupa uno spazio piuttosto ampio di memoria ed è molto facile esaurire la memoria disponibile se vengono utilizzate troppe dimensioni. Di seguito viene fornita una lista dei numeri di byte occupati da una matrice:

5 byte per il nome della matrice
2 byte per ciascuna dimensione
2 byte/elementi per le variabili intere
5 byte/elementi per le variabili numeriche normali
3 byte/elementi per le variabili stringa
1 byte per ogni carattere in ogni elemento di stringa

Le matrici intere occupano due quinti di spazio rispetto alle matrici in virgola mobile (es. DIM A%(100) richiede 209 byte; DIM A(100) richiede 512 byte).

NOTA: Gli elementi sono numerati da 0. Ad esempio DIM A(100) dà 101 elementi.

In un'istruzione DIM può essere dimensionata più di una matrice separando con virgole il nome variabile di matrice. Se il programma esegue più di una volta un'istruzione DIM per ogni matrice, verrà visualizzato il messaggio "RE'DIM ARRAY ERROR". Si consiglia di introdurre le istruzioni DIM all'inizio del programma.

ESEMPIO:

10 DIM A$(40),B7(15),CC%(4,4,4)

41 elementi, 16 elementi, 125 elementi

DIRECTORY

– Visualizza sullo schermo il contenuto dell'elenco del disco.

DIRECTORY [Dnumero drive][,<ON|,>Udispositivo][,carattere variabile]

Il tasto funzione F3 in modalità C128 visualizza l'elenco per il numero di dispositivo 8, drive 0. Utilizzare CONTROL S o NO SCROLL per arrestare la visualizzazione; per riprendere la visualizzazione, premere un tasto qualsiasi. Il tasto COMMODORE rallenta la visualizzazione. Il comando DIRECTORY non deve essere utilizzato per stampare una copia su carta perchè alcune stampanti interferiscono con i dati inviati dal disk drive. Per poter effettuare la copia su carta, si dovrà caricare l'elenco del disco (LOAD"$,8) cancellando il programma correnttemente in memoria. Il numero di dispositivo di default è 8, ed il numero di drive di default è 0.

ESEMPI:

DIRECTORY	Lista tutti i file sul disco nell'unità 8.
DIRECTORY D1,U9,"LAVORO"	Lista il file chiamato "LAVORO" sul drive 1 dell'unità 9.
DIRECTORY "CO*"	Lista tutti i file che inizianoo con le lettere "CO" come ad esempio COMPUTER, COPIA, ecc. su tutti i drive dell'unità 8. L'asterisco specifica un carattere variabile dove tutti i file che iniziano con "CO" vengono visualizzati.
DIRECTORY D0, "FILE" ?.BAK"	Il simbolo ? è un carattere variabile che corrisponde ad un qualsiasi carattere in quella posizione, ad esempio, FILE 1.BAK, FILE 2.BAK, FILE 3.BAK.

DIRECTORY D1,U9,(A$)

Lista il nome di file memorizzato nella variabile A$ sul dispositivo 9, drive 1. Ricordare di racchiudere la variabile tra parentesi quando viene utilizzata come nome di file.

NOTA: Per stampare l'elenco del disco nel drive 0, unità 8, utilizzare l'esempio seguente:

```
LOAD"$0",8
OPEN4,4:CMD4:LIST
PRINT#4:CLOSE4
```

DLOAD

– Carica un programma BASIC dal disco

DLOAD "nomefile" [,Dnumero drive][<ON|,>Unumero dispositivo]

Questo comando carica un programma BASIC dal disco alla memoria corrente (per caricare i programmi da nastro utilizzare LOAD). Il programma deve essere specificato per mezzo di un nome di file composto da un massimo di 16 caratteri. DLOAD assume come default il dispositivo 8 ed il drive 0.

ESEMPI:

DLOAD "RECBANCO"

Cerca sul disco il programma "RECBANCO" e lo carica.

DLOAD (A$)

Carica dal disco un programma il cui nome è memorizzato nella variabile A$. Se A$ è vuota, verrà visualizzato un messaggio di errore. Ricordare di racchiudere la variabile tra parentesi se utilizzata come nome di file.

Il comando DLOAD può essere utilizzato all'interno di un programma BASIC per ricercare un altro programma sul disco. Questa operazione viene chiamata concatenamento.

DO/LOOP/WHILE/UNTIL/EXIT

– Definisce e controlla il loop di programma

DO[UNTIL condizione|WHILE condizione] istruzioni [EXIT] LOOP [UNTIL condizione|WHILE condizione]

La struttura del loop esegue le istruzioni tra l'istruzione DO e l'istruzione LOOP. Se UNTIL o WHILE non modificano l'istruzione DO o LOOP, l'esecuzione delle istruzioni continuerà all'infinito. Quando nel corpo di un loop DO viene incontrata un'istruzione EXIT, l'esecuzione sarà trasferita alla prima istruzione dopo l'istruzione LOOP. I loop DO possono essere nidificati, seguendo le regole della struttura FOR-NEXT. Se viene specificato il parametro UNTIL, il programma continuerà ad eseguire il loop finchè la condizione non si verificherà. Il parametro WHILE è fondamentalmente l'opposto del parametro UNTIL: il programma continua l'esecuzione del loop finchè non si verifica la condizione. Quando la condizione non è più vera, il controllo del programma riprende l'esecuzione dall'istruzione immediatamente successiva all'istruzione LOOP. Un esempio di condizione (argomento booleano) potrebbe essere A=1, oppure G>65.

ESEMPIO:

```
10 X = 25
20 DO UNTIL X = 0
30 X = X-1
40 PRINT "X=";X
50 LOOP
60 PRINT "Fine loop"
```

Questo esempio esegue le istruzioni X=X e PRINT "X=";X fino a quando X=0. Quando X=0, il programma riprende l'esecuzione della istruzione PRINT "Fine del loop" immediatamente successiva a LOOP.

```
10 DO WHILE A$="    ":GET A$:LOOP
20 PRINT "IL TASTO";A$;"È STATO PREMUTO"
```

A$ è nullo finchè non viene premuto alcun tasto. Non appena viene premuto un tasto, il controllo del programma viene trasferito all'istruzione immediatamente successiva a LOOP, PRINT "IL TASTO";A$;"È STATO PREMUTO". L'e-

sempio esegue GET A$ finchè A$ rimane un carattere nullo. Questo loop controlla costantemente se un tasto viene premuto da tastiera.
(**NOTA:** GETKEY A$ ha lo stesso effetto della riga 10).

```
10 DOPEN #8,"FILESEQ"
20 DO
30 GET #8,A$
40 PRINT A$;
50 LOOP UNTIL ST
60 DCLOSE #8
```

Questo programma apre il file "FILESEQ" e legge i dati finché la variabile di sisteama ST indica che tutti i dati sono stati introdotti.

DOPEN

– Apre un file disco per un'operazione di lettura e/o scrittura

DOPEN #numero file logico,"nomefile[,<S|P>]"[,Llunghezza record][,Dnumero drive][<ON|,>Unumero dispositivo][,w]

dove:

S = Tipo di file sequenziale
P = Tipo di file di programma
L = Lunghezza del record = solo la lunghezza dei record di un file relativo
W =Operazione di scrittura (se non specificato, si verifica un'operazione di lettura)

Questa istruzione apre un file sequenziale, relativo o ad accesso casuale per un'operazione di lettura o scrittura. La lunghezza del record (L) si riferisce ad un file relativo, che può avere una lunghezza massima di 254 caratteri. Il parametro "W" viene specificato solo durante un'operazione di scrittura (PRINT#) in un file sequenziale. Se non specificato altrimenti, il disk drive assume che l'operazione disco da effettuare è un'operazione di lettura. I FILE RELATIVI vengono aperti per le operazioni di lettura e scrittura contemporanee.

Il numero di file logico associa un numero al file come riferimento per le prossime operazioni disco come ad esempio le operazioni di lettura (input#) o di scrittura (print#). Il numero di file logico può essere compreso nella gamma da 1 a 255. I numeri di file logici maggiori di 128 inviano automaticamente un ritorno carrello ed un'avanzamento riga con ogni comando di scrittura (print#). I numeri di file logici minori di 128 inviano solo un ritorno carrello, che però può essere eliminato introducendo un punto e virgola alla fine del comando print#. Il dispositivo di default è il numero 8, ed il drive di default è 0.

ESEMPI:

DOPEN#1,"INDIRIZZO",W — Apre il file sequenziale 1 (INDIRIZZO) per eseguire un'operazione di scrittura.

DOPEN#2,"RICETTE",D1,U9 — Apre il file sequenziale 2 (RICETTE) per eseguire un'operazione di lettura sul dispositivo 9, drive 1.

DOPEN#3,"LIBRI",L128 — Apre il file relativo 3 (LIBRI) per eseguire operazioni di lettura o scrittura. la lunghezza del record è 128 caratteri.

DRAW

– Traccia sullo schermo punti, righe e figure alle posizioni specificate.

DRAW [sorgente colore],X1,Y1[TO X2, Y2]...

Questa istruzione traccia punti, righe e figure individuali. I valori di parametro sono:

Sorgente colore	0 = Sfondo a matrice di punti 1 = Primo piano a matrice di punti 2 = Multicolor 1 } Modalità grafiche 3 e 4 3 = Multicolor 2 }
X1,Y1	Coordinata di partenza scalata
X2,Y2	Coordinata finale scalata

Vedere anche il comando LOCATE per informazioni sul cursore pixel.

ESEMPI:

DRAW 1,100,50 Traccia un punto

DRAW , 10,10 TO 100,60 Traccia una riga.

DRAW , 10,10 TO 10,60 TO 100,60 TO 10,10
Traccia un triangolo.

È possibile omettere un parametro ma è necessario introdurre la virgola che avrebbe seguito il parametro che non è stato specificato.

DSAVE

– Salva un file di programma BASIC su disco

DSAVE "nomefile" [,Dnumero drive][<ON|,>Unumero dispositivo]

Questo comando memorizza un programma BASIC su disco (vedere SAVE per la memorizzazione dei programmi su nastro). Deve essere introdotto un nome di file composto da un massimo di 16 caratteri. Il numero del dispositivo di default è 8, mentre il numero del drive di default è 0.

ESEMPI:

DSAVE "RECBANCO"	Salva il programma "RECBANCO" su disco.
DSAVE (A$)	Salva il programma disco definito nella variabile A$.
DSAVE "PROG 3",D1,U9	Salva il programma "PROG 3" sul disco, unità 9, drive 1 (disk drive doppio).

DVERIFY

– Confronta il programma in memoria con quello sul disco

DVERIFY "nomefile"[,Dnumero drive][<ON|,>Unumero dispositivo]

Questo comando provoca il confronto da parte del Commodore 128 del programma sul drive specificato con il programma in memoria. Il numero del drive di default è 0 ed il numero del dispositivo di default è 8.

NOTA: Se un'area grafica viene allocata o disallocata dopo SAVE, si verifica un errore. Tecnicamente questa operazione è corretta. Il testo BASIC viene spostato dalla posizione originale dove è allocata o disallocata un'area grafica a matrice di punti. Per questo motivo la posizione originale in cui il C128 ha effettuato il confronto del programma salvato sarà diversa. In questo caso infatti VERIFY, che effettua il confronto byte/byte, non funziona anche se il programma è valido.

Per confrontare dati **Binari**, vedere il formato VERIFY "nomefiie",8,1 nella descrizione del comando VERIFY.

ESEMPI:

DVERIFY "C128"	Confronta il programma "C128" nel drive 0, unità 8.
DVERIFY "SPRITE",D0,U9	Confronta il programma "SPRITE" nel drive 0, dispositivo 9.

END

– Definisce la fine dell'esecuzione del programma

END

Quando il programma incontra l'istruzione END, arresta immediatamente l'esecuzione. Per riavviare il programma dalla istruzione successiva (se esistente) dopo l'istruzione END, utilizzare l'istruzione CONT.

ENVELOPE

– Definisce l'inviluppo di uno strumento musicale

ENVELOPE n[,att][,dec][,sos][,ril][,fo][,li]

dove:

n .... Numero inviluppo (0-9)
att .... Attacco (0-15)
dec .... Decadimento (0-15)
sos .... Sostegno (0-15)
ril .... Rilascio (0-15)
fo .... Forma d'onda:
0 = triangolo
1 = dente di sega
2 = impulso variabile (quadra)
3 = rumore
4 = modulazione ad anello
li .... Larghezza dell'impulso (0-4095)

Un parametro non specificato manterrà il valore predefinito o correntemente ridefinito. La larghezza dell'impulso si riferisce solo alla larghiezza della forma d'onda ad impulso variabile (fo=2) e viene determinata dal rapporto ai /40,95. Ad esempio ai =2048 produce un'onda quadra, mentre 0 e 4095 producono un'uscita c.c. costante. Il Commodore 128 possiede i seguenti 10 inviluppi

iniziali:

	n	A	D	S	R	fo	li	strumento
INVILUPPO	0,	0,	9,	0,	0,	2,	1536	piano
INVILUPPO	1,	12,	0,	12,	0,	1		fisarmonica
INVILUPPO	2,	0,	0,	15,	0,	0,		calliope
INVILUPPO	3,	0,	5,	5,	0,	3		tamburo
INVILUPPO	4,	9,	4,	4,	0,	0,		flauto
INVILUPPO	5,	0,	9,	2,	1,	1,		chitarra
INVILUPPO	6,	0,	9,	0,	0,	2,	512	clavicembalo
INVILUPPO	7,	0,	9,	9,	0,	2,	2048	organo
INVILUPPO	8,	8,	9,	4,	1,	2,	512	tromba
INVILUPPO	9,	0,	9,	0,	0,	0,		xilofono

Per suonare inviluppi predefiniti di strumenti musicali specificare semplicemente il numero di inviluppo nel comando PLAY (vedere PLAY). Non è necessario utilizzare il comando ENVELOPE in quanto questo comando viene usato per modificare l'inviluppo.

FAST

– Attiva la modalità operativa a 2 MHz

FAST

Questo comando imposta la modalità a 2 MHz, provocando la disattivazione dello schermo a 40 colonne per il VIC. Tutte le operazioni (ad eccezione dell'input/output) vengono notevolmente accelerate. Può essere utilizzata la modalità grafica, che sarà visibile solo quando sarà emesso un comando SLOW.

FETCH

– Preleva i dati dalla memoria di espansione (modulo RAM)

FETCH#byte, iniz,inesp,nbes

dove:

Byte = numero dei byte da prelevare nella memoria di espansione (0-65535)
iniz = indirizzo iniziale della ram host (0-65535)
nbes = numero di banco della RAM di espansione a 64K (0-15)
inesp = indirizzo iniziale della RAM di espansione (0-65535)

FILTER

– Definisce i parametri sonori del filtro (chip SID)

FILTER [freq][,pb][,pbd][,pa][,ris]

dove:

freq .... Frequenza di taglio del filtro (0-2047)

pb .... Attivazione (1) o disattivazione (0) del filtro passa-basso

pbd .... Attivazione (1) o disattivazione (0) del filtro passa-banda

pa .... Attivazione (1) o disattivazione (0) del filtro passa-alto

ris .... Risonanza (0-15)

Se i parametri non vengono specificati il valore corrente non sarà modificato.

È possibile utilizzare più di un filtro alla volta. Ad esempio, i filtri passa-basso e passa-alto possono essere utilizzati insieme per produrre un blocco di banda a ripidità di fronte. Per ottenere un effetto udibile del filtro, occorre selezionare almeno un tipo di filtro attraverso il quale deve passare almeno una voce.

ESEMPI:

FILTER 1024,0,1,0,2 — Imposta la frequenza di taglio a 1024, seleziona il filtro passa-banda ed un livello di risonanza 2.

FILTER 2000,1,0,1,10 — Imposta la frequenza di taglio a 2000, seleziona i filtri passa-basso e passa-alto (per creare un blocco di banda a ripidità di fronte) ed imposta il livello di risonanza a 10.

FOR/TO/STEP/NEXT

– Definisce una struttura di programma a loop ripetitivo

FOR variabile = valore iniziale TO valore finale [STEP incremento]

Questa istruzione insieme all'istruzione NEXT permette di ripetere una sezione di programma per un numero di volte specificato (es. un loop). Questa funzione è utile quando è necessario effettuare un conteggio o quando un'operazione deve essere eseguita un dato numero di volte (es. una stampa).

Questa istruzione esegue ripetutamente tutti i comandi che si trovano tra le istruzioni FOR e NEXT a seconda sei valori iniziali e finali impostati. Il valore iniziale ed il valore finale rappresentano l'inizio e la fine del conteggio per la variabile loop. La variabile loop viene aggiunta o sottratta nel corso del loop FOR/NEXT.

La logica dell'istruzione FOR/NEXT è la seguente. Per prima cosa la variabile loop viene impostata al valore iniziale.

Quando il programma raggiunge una riga di programma contenente l'istruzione NEXT, viene aggiunto l'incremento STEP (default =1)al valore della variabile loop e viene effettuato un controllo per verificare se la variabile è maggiore o uguale al valore finale del loop. Se questa è inferiore al valore finale, il loop verrà eseguito nuovamente a partire dall'istruzione immediatamente successiva all'istruzione FOR. Se la variabile loop è maggiore del valore finale, il loop termina ed il programma viene ripreso a partire dall'istruzione immediatamente successiva a NEXT. Se il valore di step è negativo, avviene l'esatto contrario.

ESEMPIO (A):

```
10 FOR L = 1 TO 10
20 PRINT L
30 NEXT L
40 PRINT"FINITO! L = ";L
```

ESEMPIO (B):

```
10 FOR L=10 TO 1 STEP-1
20 PRINT L
30 NEXT L
40 PRINT "FINITO! L=";L
```

Il programma (A) stampa i numeri da 1 a 10 seguiti dal messaggio FINITO! L = 11. Il programma (B) stampa i numeri da 10 a 1 e quindi FINITO! L=0.

Il valore finale del loop può essere seguito dalla parola STEP e da un altro numero o variabile. In questo caso il valore successivo a STEP viene aggiunto ogni volta invece che una sola volta. Questo

permette il conteggio a ritroso, il conteggio per frazione o per incrementi maggiori di uno.

L'utente può impostare loop all'interno di altri loop. Questa funzione viene chiamata nidificazione di loop. Quando si utilizza la nidificazione di loop, l'ultimo loop da iniziare sarà il primo loop da finire.

ESEMPIO:

```
10 FOR L = 1 TO 100
20 FOR A = 5 TO 11 STEP .5
30 NEXT A
40 NEXT L
```

Il loop FOR...NEXT nelle righe 20 e 30 è nidificato nel loop delle righe 10 e 40. L'incremento STEP di 0.5 viene utilizzato per mostrare come siano possibili gli indici in virgola mobile. Vedere anche l'istruzione NEXT.

GET

– Riceve i dati di input dalla tastiera, un carattere per volta, senza attendere la pressione di un tasto

GET lista variabili

L'istruzione GET legge tutti i caratteri battuti dall'utente. Quando l'utente batte, i caratteri vengono memorizzati nella memoria del computer (in un'area chiamata buffer della tastiera). In questa area vengono memorizzati 10 caratteri al massimo, e tutti i caratteri dopo il decimo vengono persi. L'istruzione GET legge il primo carattere dal buffer e sposta il resto verso l'alto liberando la memoria. La parola GET è seguita da un nome di variabile, numerica o stringa. Se nel buffer non sono presenti caratteri, verrà ritornato un carattere nullo (vuoto).

GET non effettuerà una pausa nel programma, se nel buffer non vi è alcun carattere (vedere GETKEY).

Se il C128 attende la pressione di un tasto numerico o viene premuto un tasto non numerico, il programma si arresterà e verrà visualizzato un messaggio di errore. L'istruzione GET può inoltre essere inserita in un loop per la verifica di un risultato nullo. In questo caso può essere utilizzata anche l'istruzione GETKEY. Vedere GETKEY per ulteriori dettagli. Le istruzioni GET e GETKEY possono essere eseguite solo all'interno di un programma.

ESEMPIO:

10 DO:GETA$:LOOP UNTIL A$="A" — Questa riga attende la pressione del tasto A per poter continuare.

20 GET B,C,D — Riceve le variabili numeriche B,C,D da tastiera senza attendere la pressione di un tasto.

GETKEY

– Riceve idati di input da tastiera, un carattere per volta, ed attende la pressione di un tasto

GETKEY lista variabili

L'istruzione GETKEY è molto simile all'istruzione GET. Al contrario dell'istruzione GET però, GETKEY attende la pressione di un tasto da tastiera da parte dell'utente se non è presente alcun carattere nel buffer della tastiera. Questo permette al computer di attendere la battitura di un singolo carattere. Questa istruzione può essere eseguita solo all'interno di un programma.

ESEMPIO:

10 GETKEY A$ — Questa riga attende la pressione di un tasto per continuare il programma.

10 GETKEY A$,B$,C$ — Questa riga attende l'introduzione da tastiera di tre caratteri alfanumerici. GETKEY può essere utilizzato anche per i tasti numerici.

NOTA: GETKEY non può ritornare caratteri nulli (vuoti).

GET#

– Riceve i dati di input da nastro, disco o RS232

GET# numero file, lista variabili

Questa istruzione introduce un carattere alla volta da un file aperto precedentemente. Negli altri casi funziona come l'istruzione GET. Questa istruzione può essere eseguita solo all'interno di un programma.

ESEMPIO:

**10 GET#1,A$** — Questo esempio riceve un carattere, memorizzato nella variabile A$, dal numero di file 1. Questo esempio assume che il file 1 è stato aperto precedentemente. Vedere l'istruzione OPEN/DOPEN.

GO64

– Attiva la modalità C64

GO64

Questa istruzione passa dalla modalità C128 alla modalità C64. Sullo schermo viene visualizzata la domanda "Are You Sure?" in risposta all'istruzione GO64. Se viene battuto Y il programma correntemente in memoria verrà perso ed il controllo sarà passato alla modalità C64; in caso contrario, premendo un qualsiasi altro tasto il computer rimarrà nella modalità C128. Questa istruzione può essere utilizzata in modalità diretta o all'interno di un programma. Il prompt non viene visualizzato in modalità programma.

GOSUB

– Chiama una subroutine dal numero di riga specificato

GOSUB numero riga

Questa istruzione è simile all'istruzione GOTO ad eccezione che il Commodore 128 ritorna al punto di partenza al termine della subroutine. Quando viene incontrata una riga contenente un'istruzione RETURN, il programma tornerà all'istruzione immediatamente successiva all'istruzione GOSUB.

Il rimando di un'istruzione GOSUB viene chiamato subroutine. Una subroutine è utile se si deve ripetere un'operazione particolare diverse volte all'interno di un programma. Invece di duplicare ripetutamente una parte di programma, impostare una subroutine ed utilizzare un'istruzione GOSUB in un punto appropriato del programma. Vedere anche l'istruzione RETURN.

ESEMPIO:

```
20 GOSUB 800
:
:
799 END
800 PRINT "CIAO":RETURN
```

Questo esempio richiama la subroutine che inizia nella riga 800 e la segue. Tutte le subroutine devono terminare con un'istruzione RETURN.

La riga 799 arresta il programma che altrimenti eseguirebbe la subroutine.

GOTO/GO TO

– Trasferisce l'esecuzione del programma al numero di riga specificato

GOTO numero riga

Quando il computer incontra un'istruzione GOTO all'interno di un programma, esegue l'istruzione specificata dal numero di riga nell'istruzione GOTO. Se utilizzato in modalità diretta, GOTO esegue il programma a partire dal numero di riga specificato senza cancellare le variabili. Questa operazione è simile al comando RUN, ad eccezione del fatto che RUN non cancella i valori di variabile.

ESEMPI:

```
10 PRINT"COMMODORE"
20 GOTO 10
```

Il GOTO nella riga 20 causa la ripetizione continua della riga 10 fino alla pressione di RUN/STOP.

```
GOTO 100
```

Avvia l'esecuzione del programma a partire dalla riga 100, senza cancellare l'area di memorizzazione delle variabili.

GRAPHIC

– Seleziona una modalità grafica

GRAPHIC modalità [,cancellazione][,s]

oppure

GRAPHIC CLR

Questa istruzione imposta il Commodore 128 su una delle sei

modalità grafiche:

modalità	Descrizione
0	testo a 40 colonne
1	grafica a matrice di punti standard
2	grafica a matrice di punti standard (schermo diviso)
3	grafica a matrice di punti multicolore
4	grafica a matrice di punti multicolore (schermo diviso)
5	testo a 80 colonne

Il parametro di cancellazione specifica se lo schermo a matrice di punti viene cancellato (uguale a 1) all'esecuzione del programma oppure se viene lasciato intatto (uguale a 0). Il parametro S indica il numero di riga iniziale dello schermo in caso di modalità grafica 2 o 4 (modalità a schermo diviso a matrice di punti standard o multicolore). Il numero iniziale di riga di default dello schermo diviso è 19.

Quando viene eseguita, l'istruzione GRAPHIC 1-4 crea un'area a matrice di punti a 9K. L'inizio dell'area di testo BASIC viene spostata sopra l'area a matrice di punti e tutti i programmi BASIC vengono riallocati automaticamente. Questa area rimane allocata anche se l'utente torna alla modalità TESTO (GRAPHIC 0). Se viene specificato il valore 1 per l'opzione di cancellazione, lo schermo viene cancellato. Il comando GRAPHIC CLR disalloca l'area a matrice di punti a 9K, e la rende nuovamente disponibile per il testo BASIC. Qualsiasi programma BASIC viene riallocato.

ESEMPI:

GRAPHIC 1,1 — Seleziona una modalità a matrice di punti standard e cancella la matrice di punti.

GRAPHIC 4,0,10 — Seleziona la modalità a matrice di punti multicolore a schermo diviso, non cancella la matrice di punti e inizia lo schermo diviso alla riga 10.

GRAPHIC 0 — Seleziona il testo a 40 colonne.

GRAPHIC 5 — Seleziona il testo a 80 colonne.

GRAPHIC CLR — Cancella e disalloca lo schermo a matrice di punti.

HEADER

– Formatta un dischetto

**HEADER "nomedisco"[,I i.d.][,Dnumero drive]
[<ON=,>Unumero dispositivo]**

dove:

nomedisco — qualsiasi nome composto da un massimo di 16 caratteri.

i.d. — due caratteri alfanumerici qualsiasi, escluso spazi.

Prima di utilizzare per la prima volta un disco nuovo, occorre formattarlo per mezzo del comando HEADER. Questo comando può essere utilizzato anche per cancellare un disco precedentemente formattato che in questo modo può essere riutilizzato.

Introducendo un comando HEADER in modalità diretta, apparirà il messaggio **ARE YOU SURE?** In modalità programma il prompt non viene visualizzato.

Questo comando divide il disco in sezioni chiamate blocchi e crea un indice di file chiamato elenco. Ad ogni disco deve essere assegnato un numero i.d. esclusivo. Prestare attenzione nell'uso di HEADER in quanto questo comando cancella tutti i dati memorizzati.

Se un dischetto è già stato formattato, il comando HEADER può essere utilizzato in modo più veloce omettendo l'i.d. che si introduce per un nuovo disco; infatti verrà utilizzato il vecchio i.d. Questa operazione può essere effettuata solo se il disco è stato precedentemente formattato in quanto verrà solamente cancellato l'elenco e il disco non sarà formattato. Il numero di dispositivo di default è 8 ed il drive di default è 0.

Per conferma il sistema visualizza "ARE YOU SURE?" prima del completamento dell'operazione da parte del Commodore 128. Premere il tasto "Y" per confermare, oppure premere un qualsiasi altro tasto per annullare l'operazione.

Il comando HEADER legge il canale di errore del comando disco e, se viene riscontrato un errore, appare il messaggio "?BAD DISK".

Il comando HEADER è analogo al comando del BASIC 2.0:

OPEN 1,8,15,"N0:nomedisco,i.d."

ESEMPI:

HEADER "MIODISCO",I51,D0 Formatta "MIODISCO" utilizzando l'i.d. 51 sul drive 0, numero di dispositivo (di default) 8.

HEADER "REC",I45,D1 ON U9 Formatta "REC" utilizzando l'i.d. 45, sul drive 1, numero di dispositivo 9.

HEADER "PROGRAMMI C128",D0
Operazione di formattazione veloce sul drive 0, numero dispositivo 8, in cui il disco era già stato formattato. Viene utilizzato il vecchio numero i.d.

NOTA: Non è possibile utilizzare una variabile stringa come i.d.

HELP

– Evidenzia la riga in cui è stato riscontrato l'errore

HELP

Il comando HELP viene utilizzato dopo l'individuazione di un errore all'interno di un programma. Quando HELP viene battuto nel formato a 40 colonne, viene listata la riga in cui si è verificato l'errore, con la parte contenente l'errore visualizzata in schermo invertito. Nel formato a 80 colonne la parte di riga in cui si è verificato l'errore viene sottolineata. Premendo il tasto HELP si otterrà la battitura di HELP ed il ritorno a capo automatico.

IF/THEN/ELSE

– Esegue il calcolo di un'espressione condizionale ed esegue le parti di un programma che dipendono dal risultato dell'espressione.

IF espressione THEN istruzioni BASIC 2.0
IF espressione THEN istruzioni [:ELSE clausola-else] BASIC 7.0

L'istruzione IF...THEN valuta un'espressione BASIC ed effettua una delle due possibili operazioni a seconda del risultato dell'espressione. Se l'espressione è vera, l'istruzione(i) successiva a THEN viene eseguita (una qualsiasi istruzione BASIC). Se l'espressione è falsa, il programma riprenderà l'esecuzione dalla riga di programma immediatamente successiva alla riga contenente l'istruzione IF, a meno che non sia presente una clausola ELSE. L'intera istruzione IF...THEN deve essere lunga al massimo 160 caratteri (80 in modalità C64). Vedere anche BEGIN/BEND.

La clausola ELSE, se presente, deve trovarsi sulla stessa riga della parte IF...THEN dell'istruzione e deve essere separata dalla clausola THEN da un simbolo di due punti (:). Se è presente una clausola ELSE, verrà eseguita solo quando l'espressione è falsa. L'espressione calcolata può essere una variabile o una formula e sarà considerata vera se non sarà zero, oppure falsa se zero. Nella maggior parte dei casi è presente un'espressione che coinvolge operatori relazionali (=, <, >,>=, <=,<>).

L'istruzione IF...THEN può essere espressa in due forme:

IF espressione THEN numero riga
oppure:
IF espressione GOTO numero riga

Queste due forme trasferiscono l'esecuzione del programma al numero di riga specificato se l'espressione è vera. In caso contrario, il programma riprenderà l'esecuzione a partire dal numero di riga immediatamente successiva alla riga contenente l'istruzione IF:

ESEMPIO:

50 IF X > 0 THEN PRINT "OK":ELSE END

Questa riga controlla il valore di X. Se X è maggiore di 0, l'istruzione immediatamente successiva alla parola chiave THEN (PRINT "OK") viene eseguita e la clausola ELSE ignorata. Se X è minore o uguale a 0, la clausola ELSE viene eseguita e l'istruzione immediatamente successiva a THEN Y viene ignorata.

```
10 IF X = 10 THEN 100
20 PRINT "X non è uguale a 10"
:
99 STOP
100 PRINT "X è uguale a 10"
```

Questo esempio calcola il valore di X. Se X è uguale a 10, il controllo del programma viene trasferito alla riga 100 e viene visualizzato il messaggio "X è uguale a 10". Se X non è uguale a 10, il programma riprende dalla riga 20, il C128 stampa il messaggio "X non è uguale a 10" ed il programma si arresta.

NOTA: La clausola ELSE non può essere utilizzata in modalità C64.

INPUT

– Riceve una stringa dati o un numero dalla tastiera ed attende la pressione di RETURN da parte dell'utente

INPUT ["stringa prompt";] lista variabili

L'istruzione INPUT chiede l'introduzione di dati nel corso dell'esecuzione del programma ed inserisce i dati in una variabile o variabili. Il programma si arresta, visualizza un punto di domanda (?) sullo schermo ed attende l'introduzione della risposta e la pressione del tasto RETURN. La parola INPUT è seguita da una stringa prompt e da un nome di variabile o lista di nomi di variabili separati da virgole. Il messaggio nella stringa prompt all'interno delle virgolette visualizza le informazioni da introdurre da parte dell'utente. Se viene introdotto questo messaggio, dopo le virgolette di chiusura del prompt deve essere presente un punto e virgola (;).

Quando vengono introdotte più variabili, occorre separarle con virgole. Il computer chiede di introdurre i valori rimanenti visualizzando due punti interrogativi (??). Se viene premuto il tasto RETURN senza introdurre un valore, la variabile di INPUT mantiene il valore introdotto precedentemente. L'istruzione INPUT può essere eseguita solo all'interno di un programma.

ESEMPIO:

```
10 INPUT "INTRODUCI UN NUMERO";A
20 INPUT "ED IL TUO NOME";A$
30 PRINT A$;"HAI INTRODOTTO IL NUMERO";A
```

INPUT#

– Introduce nella memoria del computer i dati di un file

INPUT# numero file, lista variabili

Questa istruzione funziona come INPUT, ma preleva i dati di un file precedentemente aperto, generalmente su disco o nastro invece che da tastiera. Non viene utilizzata alcuna stringa prompt. Questa istruzione può essere utilizzata solo all'interno di un programma.

ESEMPIO:

```
10 OPEN 2,8,2,"FILEDATI,S,R"
20 INPUT #2,A$,C,D$
```

Questa istruzione introduce i dati dal file FILEDATI e li memorizza nelle variabili A$, C e D$.

KEY

– Definisce o lista gli assegnamenti dei tasti funzione

KEY [numero tasto, stringa]

Sul Commodore 128 sono presenti otto tasti funzione (F1-F8): quattro vengono battuti direttamente da tastiera e quattro vengono battuti contemporaneamente a SHIFT. Il Commodore 128 permette di eseguire una funzione o un'operazione ogni volta che il tasto funzione specifico viene premuto. La definizione assegnata ad un tasto può consistere di dati, oppure di un comando o serie di comandi. KEY senza la specifica di alcun parametro ritorna un listato che visualizza tutti gli assegnamenti correnti di KEY. Se ad un tasto funzione vengono assegnati dei dati, questi vengono visualizzati sullo schermo alla pressione del tasto funzione relativo. La lunghezza massima delle definizioni totali è 246 caratteri.

ESEMPIO:

KEY 7,"GRAPHIC0" + CHR$(13) + "LIST" + CHR$(13)

Questo ordina al computer di selezionare lo schermo testo (VIC) a 40 colonne e di listare il programma ogni volta che viene premuto il tasto F7 (in modalità diretta). CHR$(13) è il carattere ASCII per RETURN ed ha lo stesso effetto della pressione del tasto RETURN. Utilizzare CHR$(27) per ESC. Utilizzare CHR$(34) per incorporare il carattere di virgolette in una stringa KEY. In un programma i tasti possono essere ridefiniti. Ad esempio:

10 KEY 2,"PRINT DS$"+ CHR$(13)

Questo ordina al computer di controllare e visualizzare le variabili di canale di errore del disk drive (PRINT DS$) ogni volta che viene premuto il tasto funzione F2.

Per riportare tutti i tasti funzione ai valori default BASIC, reimpostare il Commodore 128 premendo il pulsante RESET.

LET

– Assegna un valore ad una variabile

[LET] variabile = espressione

La parola LET viene usata raramente nei programmi in quanto non è necessaria. Quando viene definita una variabile o quando viene assegnato un valore ad una variabile, LET è sempre sottinteso. Il nome di variabile che riceve il risultato di un calcolo è a sinistra del segno di uguale mentre il numero, la stringa o la formula sono a destra. Con un'istruzione LET (sottintesa) è possibile solo assegnare un valore. Ad esempio, LET A = B = 2 è illegale.

ESEMPIO:

10 LET A = 5	Assegna il valore 5 alla variabile numerica A.
20 B = 6	Assegna il valore 6 alla variabile numerica B.
30 C = A * B + 3	Assegna alla variabile numerica C il valore risultante da 6 per 5 più 3.
40 D$ = "CIAO"	Assegna la stringa "CIAO" alla variabile stringa D$.

LIST

– Lista il programma BASIC correntemente in memoria

LIST[riga|prima-|prima-ultima|-ultima]

Il comando LIST visualizza un listato di programma BASIC che è stato battuto o caricato nella memoria del Commodore 128 in modo che lo si possa leggere e modificare. Quando LIST viene utilizzato da solo (non seguito da numeri), il Commodore 128 visualizza sullo schermo un listato completo del programma. Questo processo può essere rallentato tenendo premuto il tasto Commodore contemporaneamente a CONTROL S o NO SCROLL, e potrà essere ripreso premendo un tasto qualsiasi. Per arrestare il processo, premere il tasto RUN/STOP. Se la parola LIST è seguita da un numero di riga, il Commodore 128 visualizzerà solo quella riga. Se LIST viene battuto con due numeri separati da un trattino, verranno visualizzate tutte le righe a partire dal primo numero indicato fino all'altro numero compreso. Se LIST viene battuto seguito da un numero ed un trattino, il Commodore 128 visualizzerà tutte le righe a partire da quel numero fino alla fine del programma. Se infine LIST viene battuto seguito da un trattino e quindi da un numero, verranno visualizzate tutte le righe dall'inizio del programma fino al numero di riga indicato. Utilizzando queste variazioni, sarà possibile visualizzare sullo schermo qualsiasi parte di programma per un eventuale controllo o modifica. Nella modalità Commodore 128, LIST può essere utilizzato all'interno di un programma.

ESEMPI:

LIST	Visualizza il programma intero.
LIST 100-	Visualizza dalla riga 100 alla fine del programma.
LIST 10	Visualizza solo la riga 10
LIST - 100	Visualizza tutte le righe dall'inizio del programma fino alla riga 100 compresa.
LIST 10-200	Visualizza le righe dalla 10 alla 200 compresa.

LOAD

– Carica un programma da un dispositivo periferico come ad esempio un disk drive o un Datassette

LOAD["nomefile"][,numero dispositivo][,flag di rilocazione]

Questo comando viene utilizzato per richiamare un programma memorizzato su disco o su nastro. Nomefile è un nome di programma composto da un massimo di 16 caratteri racchiuso tra virgolette. Il nome deve essere seguito da una virgola (al di fuori delle virgolette) e da un numero che rappresenta il numero di dispositivo su cui è memorizzato il programma (disco o nastro). Se non viene introdotto alcun numero, il Commodore 128 caricherà dal numero di dispositivo 1 (registratore Datassette).

Il flag di rilocazione è un numero (0 oppure 1) che determina dove il programma è caricato in memoria. Un flag di rilocazione 0 ordina al Commodore 128 di caricare il programma all'inizio dell'area di programma BASIC. Un flag 1 ordina al computer di caricare dal punto in cui il programma era stato salvato. Il valore di default del flag di rilocazione è 0. Il parametro di programma 1 viene utilizzato generalmente in fase di caricamento di programmi in linguaggio macchina.

Il dispositivo più comunemente utilizzato con il comando LOAD è il disk drive (numero di dispositivo 8) sebbene sia più comodo il comando DLOAD quando si lavora con i dischi.

Se LOAD viene battuto senza argomenti e seguito da RETURN, il C128 assumerà che si sta caricando da nastro, quindi visualizzerà "PRESS PLAY ON TAPE". Premendo PLAY, il Commodore 128 inizierà la ricerca del programma sul nastro. Al ritrovamento del programma, il Commodore 128 visualizzerà FOUND "nomefile", dove nomefile è il nome del primo file trovato sul nastro dal Datassette. Premere il tasto Commodore se si tratta del file desiderato, oppure premere la barra spaziatrice per continuare la ricerca sul nastro. Dopo il caricamento del programma, questo potrà essere eseguito, listato o modificato.

NOTA: Premendo la barra spaziatrice in modalità C64 il file seguente non verrà ricercato.

ESEMPI:

LOAD	Legge il programma successivo sul nastro.
LOAD "CIAO"	Ricerca sul nastro un programma chiamato CIAO e lo carica, se presente.
LOAD A$,8	Carica dal disco il programma cui nome è memorizzato nella variabile A$. Questo equivale a DLOAD (A$).
LOAD"CIAO",8	Ricerca il programma chiamato CIAO sul numero di disk drive 8, drive 0. (Equivalente di DLOAD "CIAO").
LOAD"LINGMAC",8,1	Carica il programma in linguaggio macchina chiamato "LINGMAC" nella locazione da cui era stato salvato.

Il comando LOAD può essere utilizzato all'interno di un programma BASIC per ricercare ed eseguire il prossimo programma su nastro o disco. Questa operazione viene chiamata concatenamento.

LOCATE

– Posiziona sullo schermo il cursore pixel a matrice di punti

LOCATE x,y

L'istruzione LOCATE posiziona il cursore pixel (CP) su una coordinata pixel specificata dello schermo.

Il curosore pixel (CP) è la coordinata dello schermo a matrice di punti in cui inizia il tracciamento di cerchi, riquadri, linee e punti e in cui inizia la colorazione. Il CP si muove sulle coordinate X e Y, dal valore 0,0 al valore 319,199 (scalato). Il CP non è visibile come il cursore di testo, ma può essere comandato per mezzo delle istruzioni grafiche (BOX, CIRCLE, DRAW,ecc.). La locazione di default del cursore pixel è la coordinata specificata come parte X e Y in ogni specifico comando grafico. In questo modo non è necessario specificare il comando LOCATE.

ESEMPIO:

LOCATE 160,100	Posiziona il CP al centro dello schermo a matrice di punti. Non si vedrà nulla finchè non sarà tracciato qualcosa.

Il CP può essere ottenuto utilizzando la funzione RDOT(0) per avere la coordinata X e RDOT(1) per avere la coordinata Y. La sorgente di colore del punto del CP può essere ottenuta tramite la stampa di RDOT(2).

MONITOR

– Attiva il monitor del linguaggio macchina del Commodore 128

MONITOR

Vedere l'Appendice J per informazioni sul monitor del linguaggio macchina del Commodore 128.

MOVSPR

– Posiziona o sposta gli sprite sullo schermo

MOVSPR numero,x1,y1	Posiziona lo sprite specificato alla coordinata assoluta x,y (scalata).
MOVSPR numero +|-x, |-y	Sposta lo sprite relativo alla sua posizione corrente.
MOVSPR numero,x;y	Sposta lo sprite alla distanza x, angolo y in relazione alla posizione corrente.
MOVSPR numero, angolo x# velocità y	Sposta lo sprite all'angolo relativo alle coordinate originali, in senso orario e alla velocità specificata.

dove:

numero è il numero dello sprite (da 1 a 8)
<**,x1,y1**> è la coordinata della posizione dello sprite (scalata).

ANGOLO è l'angolo (0-360) di movimento in senso orario rispetto alla coordinata originale degli sprite.

VELOCITÀ è la velocità (0-15) di movimento dello sprite.

Questa istruzione posiziona uno sprite alla locazione specifica dello schermo a seconda dello schema delle coordinate dello sprite (non lo schema della matrice di punti) o inizializza il movimento dello sprite alla velocità specificata. Vedere MOVSPR nella Sezione 6 per una spegazione completa del sistema di coordinate sprite.

ESEMPI:

MOVSPR 1,150,150	Posiziona lo sprite 1 vicino al centro dello schermo, coordinate x,y: 150,150.
MOVSPR 1,+20,–30	Sposta lo sprite 1 verso destra di 20 coordinate e verso l'alto di 30 coordinate.
MOVSPR 4,–50,+100	Sposta lo sprite 4 a sinistra di 50 coordinate e verso il basso di 100 coordinate.
MOVSPR 5,45 #15	Sposta lo sprite 5 di un angolo di 45 gradi in senso orario, in relazione alle sue coordinate x e y originali. Lo sprite si sposta alla velocità maggiore (15).

NOTA: Dopo aver specificato l'angolo e la velocità nel terzo formato dell'istruzione MOVSPR, si dovrà impostare la velocità zero per arrestare il movimento degli sprite.

NEW

– Cancella i programmi e le variabili in memoria

NEW

Questo comando cancella l'intero programma presente in memoria e cancella tutte le variabili utilizzate. Se il programma non era stato precedentemente memorizzato su disco o nastro, con questa operazione sarà perso. Usare questo comando con cautela. Il comando NEW può essere utilizzato anche come istruzione in un programma BASIC. Comunque, quando il Commodore 128 raggiunge la riga di programma in cui è contenuto NEW il programma viene cancellato e tutto si arresta.

ON

– Salta condizionatamente alla riga di programma specificata a seconda dei risultati dell'espressione specificata

ON espressione <GOTO|GOSUB> riga #1 [,riga #2,...]

Questa istruzione permette alle istruzioni GOTO e GOSUB di funzionare come versioni speciali dell'istruzione IF (condizionale). La parola ON è seguita da un'espressione quindi da GOTO o GOSUB e da una lista di numeri di riga separati da virgole. Se il risultato dell'espressione è 1, verrà eseguita la prima riga della lista. Se il risultato è 2, verrà eseguita la seconda riga e così via. Se il risultato è 0 o maggiore del numero delle righe della lista, il programma riprenderà l'esecuzione dall'istruzione immediatamente successiva all'istruzione ON. Se il numero è negativo, verrà visualizzato un messaggio ILLEGAL QUANTITY ERROR.

ESEMPIO:

```
10 INPUT X:IF X<0 THEN 10
20 ON X GOSUB 30, 40, 50, 60
25 GOTO 10
30 PRINT "X = 1":RETURN
30 PRINT "X = 2":RETURN
30 PRINT "X = 3":RETURN
30 PRINT "X = 4":RETURN
```

Quando X=1, ON invia il comando al primo numero di riga della lista (30). Quando X=2, ON invia il comando alla seconda riga (40), ecc.

OPEN

– Apre i file per input o output

OPEN numero file logico, numero dispositivo [,indirizzo secondario][<,"nomefile, tipofile, modalità"|,stringa cmd>]

L'istruzione OPEN permette al Commodore 128 di accedere ai file nei dispositivi come disk drive, registratore Datassette, stampante o schermo del Commodore 128. La parola OPEN è seguita da un numero di file logico, a cui faranno riferimento tutte le altre istruzioni BASIC di input/output, come ad esempio le istruzioni PRINT# (scrittura), INPUT# (lettura) ecc. Il numero è compreso nella gamma da 1 a 255.

Il secondo numero, chiamato numero dispositivo, segue il numero del file logico. Il numero di dispositivo 0 è la tastiera del Commodore 128; 1 è il registratore; 3 è lo schermo del Commodore 128; 4-7 è la stampante(i); e 8-11 sono i disk drive. Si consiglia di assegnare al file lo stesso numero del dispositivo in modo che possa essere facilmente ricordato.

Subito dopo il numero dispositivo è presente un terzo parametro chiamato indirizzo secondario. In caso si utilizzi una cassetta, l'indirizzo potrebbe essere 0 per lettura, 1 per scrittura e 2 per scrittura con l'indicatore di FINE NASTRO alla fine. In caso si utilizzi un disco, il numero si riferirà al numero del canale. Vedere il manuale del disk drive per ulteriori informazioni sui canali ed i loro numeri. Per quanto riguarda la stampante, gli indirizzi secondari vengono utilizzati per la selezione di alcune funzioni di programmazione.

Subito dopo l'indirizzo secondario è possibile specificare un nomefile per l'utilizzo di disco o nastro OPPURE una stringa, che può essere un comando per il disk drive o l'unità a nastro oppure il nome del file su nastro o disco. Se viene specificato il nomefile, il tipo e la modalità si riferiranno solamente ai file disco. I tipi di file sono PROGRAMMA, SEQUENZIALE, RELATIVO e UTENTE: le modalità sono LETTURA e SCRITTURA.

ESEMPI:

10 OPEN 3,3	Apre lo schermo come numero di file 3.
20 OPEN 1,0	Apre la tastiera come numero di file 1.
30 OPEN 1,1,0,"PUNTO"	Apre la cassetta per la lettura come numero di file 1, utilizzando "PUNTO" come nome di file.
OPEN 4,4	Apre la stampante come numero di file 4.
OPEN 15,8,15	Apre il canale di comando del disco come file 15, indirizzo secondario 15. L'indirizzo secondario 15 è riservato al canale di errore del disk drive.

5 OPEN 8,8,12,"PROVAFILE,SEQ,SCRIT"

Apre un file disco sequenziale per la scrittura chiamato PROVAFILE come numero di file 8, indirizzo secondario 12.

Vedere anche le istruzioni CLOSE, CMD, GET#, INPUT# e PRINT# e le variabili di sistema ST, DS e DS$.

PAINT

– Riempie di colore un'area

PAINT [sorgente colore],x,y[,modalità]

dove:

sorgente colore	0 Primo piano a matrice di punti 1 Sfondo a matrice di punti (default) 2 Multicolor 1 3 Multicolor 2
x,y	coordinata di partenza, scalata (default alla posizione del cursore pixel (CP))
modalità	0= colora un'area definita dalla sorgente di colore selezionata (default) 1= colora un'area definita da una sorgente non di sfondo

Il comando PAINT riempie un'area di colore: l'area viene definita da un perimetro completo, senza includere le coordinate X e Y specificate. Non è possibile colorare i punti in cui la sorgente di colore è la stessa di quella del cursore pixel (es. 3).

Se la modalità è = 0, l'area da riempire dovrà essere circondata dalla sorgente di colore; qualsiasi altra sorgente di colore all'interno di questo perimetro sarà "sovra-colorata" (es. 1).

Se la modalità è = 1, il perimetro della figura sarà qualsiasi sorgente di colore (tranne 0). Nessuna sorgente di colore sarà "sovra-colorata" (vale a dire che solo le aree non colorate potranno essere riempite quando la modalità è =1) (es. 2).

ESEMPIO 1:

10 COLOR 0,1:COLOR 1,2:COLOR 2,5:COLOR 3,7	
20 GRAPHIC 3,1	Grafica multicolore
30 CIRCLE 1,80,100,30	Traccia un cerchio con la sorgente di colore 1
40 CIRCLE 3,80,100,35	Traccia un cerchio con la sorgente di colore 3
50 BOX 2,80,100,90,110,45,1	Traccia un riquadro pieno con la sorgente di colore 2
60 PAINT 3,70,100,0	Colora l'interno di un cerchio con la sorgente di colore 3 racchiuso solo dalla sorgente di colore 3

ESEMPIO 2:

Come l'esempio 1, modificando la riga 60 come segue:

60 PAINT 3,70,100,1	Colora l'interno di un cerchio racchiuso da qualsiasi sorgente di colore di sfondo.

ESEMPIO 3:

Come l'esempio 2, aggiungendo le righe 70 e 80 come segue:

70 COLOR 2,8	Modifica la sorgente di colore 2 nel colore giallo.
80 PAINT 2,90,110,1	Tenta di ricolorare il riquadro (il tentativo fallisce in quanto la sorgente di colore in PAINT e (90,110) è la stessa (2)).

PLAY

– Definisce e suona note musicali ed elementi

PLAY "[Vn][On][Tn][Un][Xn][elementi][...]"

dove:

Vn= Voce (n=1-3)
On= Ottava (n=0-6)
Tn= Inviluppo del motivo (n=0-9)
0 = piano
1 = fisarmonica
2 = calliope
3 = tamburo
4 = flauto
5 = chitarra
6 = clavicembalo
7 = organo
8 = tromba
9 = xilofono
Un= Volume (n=0-9) - 0 disattivato - 9 volume totale (VOL15).
Xn= Filtro attivato (n=1), disattivato (n=0)

Elementi: **Note:** A (la), B (si), C (do), D (re), E (mi), F (fa), G (sol)

.... Diesis*
$ .... Bemolle*
W .... Intero
H .... Metà
Q .... Quarto
I .... Ottavo
S .... Sedicesimo
. ... Puntata*
R .... Pausa
M .. Attende che tutte le voci che stanno suonando al momento terminino la misura corrente.

L'istruzione PLAY permette di selezionare la voce, l'ottava e l'inviluppo del motivo (compresi dieci inviluppi predefiniti di strumenti musicali), il volume e le note desiderate. Tutti questi comandi devono essere racchiusi tra virgolette.

Tutti gli elementi tranne R e M precedono le note musicali in una stringa PLAY.

* Questi simboli devono precedere ogni nota a cui si riferiscono.

ESEMPI:

PLAY "V1O4T0U5X0CDEFGAB" Suona le note C,D,E,F,G,A e B (do,re,mi,fa,sol,la, e si) in voce 1, ottava 4, inviluppo di motivo 0 (piano), a patto che non lo si sia modificato con l'inviluppo, al volume 5, con filtro disattivato.

PLAY "V3O5T6U7X1#B$AW.CHDQEIF" Suona le note si (B) diesis, la (A) bemolle, un intero do (C) puntato, un metà re (D), un quarto mi (E) e un ottavo fa (F).

NOTA: Per poter sentire il risultato di quanto sopra, sarà necessario impostare un filtro (provare FILTER 1024,1).

POKE

– Modifica il contenuto di una locazione di memoria RAM

POKE indirizzo, valore

L'istruzione POKE permette di modificare qualsiasi valore nella RAM del Commodore 128 e permette la modifica di molti dei registratori di I/O del Commodore 128. La parola chiave POKE è sempre seguita da due parametri. Il primo è una locazione all'interno della memoria del Commodore 128 (con valori da 0 a 65535). Il secondo parametro è un valore da 0 a 255, introdotto nella locazione, che sostituisce qualsiasi valore presente in precedenza. Il valore della locazione di memoria determina la configurazione di bit della locazione di memoria. In modalità C128 POKE agisce nel banco RAM selezionato correntemente. L'indirizzo POKE dipende dal numero di banco. Vedere BANK in questa Enciclopedia per la configurazione BANK appropriata.

ESEMPIO:

10 POKE 53280,1	Modifica il colore della cornice del VIC (BANCO 15 in modalità C128)

NOTA: PEEK, una funzione relativa a POKE, la quale ritorna il contenuto della locazione di memoria specificata, viene descritta nel paragrafo FUNZIONI.

PRINT

– Emette un output allo schermo di testo

PRINT [lista di stampa]

L'istruzione PRINT è l'istruzione di output più importante nel linguaggio BASIC. Questa istruzione è la prima istruzione BASIC insegnata e ad essa possono essere applicate diverse variazioni. La parola PRINT può essere seguita da uno dei seguenti dati:

Caratteri tra virgolette	("testo")
Nomi di variabile	(A, B, A$, X$)
Funzioni	(SIN(23), ABS(33))
Simboli di punteggiatura	(;,)

I caratteri tra virgolette vengono stampati esattamente come vengono visualizzati sullo schermo. Vengono inoltre stampati i valori contenuti nei nomi di variabile (numero o stringa) ed i valori numerici delle funzioni.

I simboli di punteggiatura vengono utilizzati per la visualizzazione dei dati sullo schermo. La virgola separa i dati stampati di 10 spazi, mentre il punto e virgola li stampa uno accanto all'altro (vedere Sezione 3, stampa di numeri). Entrambi questi simboli di punteggiatura possono essere utlizzati come ultimo simbolo dell'istruzione. Si otterrà così la stampa dell'istruzione successiva esattamente come se fosse la continuazione dell'istruzione precedente.

ESEMPI:	**VISUALIZZAZIONE:**
10 PRINT "CIAO"	CIAO
20 A$="AMICO":PRINT "CIAO";A$	CIAO AMICO
30 A=4:B=2:PRINT A+B	6
40 J=41:PRINT J;:PRINT J-1	41 40
50 PRINT A;B;:D=A+B :PRINT D;A-B	4 2 6 2

Vedere anche le funzioni POS, SPC e TAB.

PRINT#

– Invia i dati ai file

PRINT#numero file, lista di stampa

Esistono alcune differenze tra questa istruzione e l'istruzione PRINT. La cosa più importante è che la parola PRINT# è seguita da un numero, che rappresenta il file dati aperto in precedenza. Il numero è seguito da una virgola e da una lista di elementi da inviare al file. Il punto e virgola funziona allo stesso modo per la spaziatura con le stampanti come nell'istruzione PRINT la virgola introdurre 10 spazi. Alcuni dispositivi non funzionano con TAB e SPC.

ESEMPIO:

10 OPEN 4,4 **20 PRINT#4,"CIAO AMICO!",** **A$,B$**	Emette "CIAO AMICO" e le variabili A$ e B$ verso la stampante.
10 OPEN 2,8,2,"FILEDATI,S,W" **20 PRINT # 2,A,B$,C,D**	Questo esempio emette le variabili dati A,B$, C e D sul file disco numero 2.

NOTA: Il comando PRINT# viene utilizzato da solo per ripristinare il canale verso un dispositivo dopo l'emissione via CMD e prima di chiudere il file, come mostrato di seguito:

```
OPEN 4,4
CMD4
PRINT#4
CLOSE4
```

Vedere anche il comando CMD.

PRINT USING

– Definisce il formato di output

PRINT [#numerofile,] USING "lista formato";lista di stampa

Questa istruzione definisce il formato degli elementi di stringa e numerici per l'emissione dei dati sullo schermo di testo, sulla stampante o altro dispositivo. Il formato viene racchiuso tra virgolette. Questa è la lista di formato. Aggiungere quindi un punto e virgola ed una lista di quanto dovrà essere stampato nel formato per la lista di stampa. La lista può essere costituita da variabili o dai valori effettivi da stampare, separati da virgole.

STRINGA DI FORMATO UTILIZZATA CON

CARATTERE	NUMERICO	STRINGA
Simbolo di numero (#)	X	X
Simbolo più (+)	X	
Simbolo meno (–)	X	
Punto decimale (.)	X	
Virgola (,)	X	
Simbolo di dollaro ($)	X	
4 accenti circonflessi (^^^^)	X	
Simbolo di uguale (=)		X
Simbolo di maggiore di (>)		X

Il simbolo di numero (#) riserva dello spazio per un singolo carattere nel campo dell'output. Con dati numerici se l'elemento dati contiene più caratteri di quanti siano i simboli # nel campo di formato, l'intero campo verrà riempito di asterischi (*) e nessun carattere verrà stampato.

ESEMPIO:

10 PRINT USING "####";X

Per questi valori di X, il formato visualizzerà:

X = 12.34	12
X = 567.89	568 (arrotondato)
X = 123456	****

Per un elemento STRINGA, i dati di stringa verranno troncati al limite del campo. Verranno stampati tanti caratteri quanti saranno i simboli di numero (#) nell'elemento di formato. Il troncamento viene eseguito a destra.

I simboli più (+) e meno (–) possono essere utilizzati come prima oppure ultima posizione di un campo di formato ma non per entrambe le posizioni. Viene stampato il simbolo più se il numero è positivo, mentre viene stampato il simbolo meno se il valore è negativo.

Se viene utilizzato un simbolo meno ed il numero è positivo, alla posizione del carattere indicata dal simbolo meno verrà stampato uno spazio vuoto.

Se non viene utilizzato nè un simbolo più nè uno meno nel campo di formato per un elemento di dato numerico, verrà stampato un simbolo meno davanti alla prima cifra, oppure un simbolo di dollaro se il numero è negativo. Non verrà stampato alcun simbolo se il numero è positivo. Questo significa che viene stampato un carattere addizionale, il simbolo meno, se il numero è negativo. Se i caratteri sono troppi da poter essere contenuti nel campo specificato dal simbolo di numero e dai simboli più/meno, si verifica un overflow ed il campo sarà riempito di asterischi (*).

Un simbolo di punto decimale (.) designa la posizione del punto decimale nel numero. In ogni campo di formato può essere presente un solo punto decimale. Se nel campo di formato non viene specificato un punto decimale, il valore sarà arrotondato al numero intero più vicino e stampato senza punto decimale.

Quando viene specificato un punto decimale, il numero delle cifre prima del punto (compreso il segno di meno, se il valore è negativo) non devono superare il numero di simboli di numero prima del punto decimale. Se sono presenti troppe cifre, si verificherà un overflow ed il campo sarà riempito di asterischi (*).

Una virgola (,) permette di introdurre virgole nei campi numerici. La posizione della virgola nella lista di formato indica il punto in cui le virgole appariranno in un numero stampato. Vengono stampate solo le virgole all'interno di un numero. Le virgole non utilizzate a sinistra della prima cifra appaiono come carattere riempitore. La prima virgola in un campo deve essere preceduta da almeno un simbolo di numero (#).

Se le virgole vengono specificate in un campo ed il numero è negativo, verrà stampato un simbolo di meno come primo carattere anche se la posizione del carattere viene specificata come virgola.

Un simbolo di dollaro ($) indica che all'interno del numero sarà stampato un simbolo di dollaro. Se questo simbolo è mobile (deve trovarsi sempre prima del numero) prima di esso dovrà essere introdotto un simbolo di numero (#). Se il simbolo di dollaro viene specificato senza un simbolo di numero iniziale, il simbolo di dollaro verrà stampato alla posizione indicata dal campo di formato. Se in un campo di formato vengono specificati un simbolo di più o di meno con un simbolo di dollaro, il programma stamperà un simbolo di più o di meno prima del simbolo di dollaro.

ESEMPI:

CAMPO	ESPRESSIONE	RISULTATO	COMMENTO
##.#	–.1	–0.1	Viene aggiunto uno zero come prima cifra.
##.#	1	1.0	Viene aggiunto uno zero finale.
####	–100.5	–101	Arrotondato per eccesso.
####	–1000	****	Si verifica un overflow in quanto le quattro cifre ed il simbolo di meno non possono essere contenuti in un campo.
###.	10	10.	Viene aggiunto un punto decimale.
#$##	1	$1	Viene aggiunto un simbolo di dollaro iniziale.

Le frecce verso l'alto, o accenti circonflessi (^^^^) vengono utilizzati per specificare che il numero è da stampare nel formato E (notazione scientifica). Per specificare l'ampiezza del campo dovrà essere utilizzato un simbolo di numero oltre ai quattro accenti circonflessi. Gli accenti circonflessi devono essere introdotti dopo il simbolo di numero in un campo di formato. Per stampare un numero nel formato E si dovranno introdurre quattro accenti circonflessi. Specificando meno di quattro accenti circonflessi si verificherà un errore di sintassi. Specificando invece più di quattro accenti, verranno presi in considerazione solo i primi quattro ed il quinto verrà interpretato come simbolo di non testo. Per centrare una stringa in un campo viene utilizzato un simbolo di uguale (=). L'ampiezza del campo viene specificata dal numero di caratteri (simboli di numero e di uguale) nel campo di formato. Se la stringa contiene meno caratteri di quanti possono essere contenuti nel campo, la stringa verrà centrata nel campo. Se la stringa contiene più caratteri di quanti ne possa contenere il campo, i caratteri all'estrema destra verranno troncati e la stringa riempirà interamente il campo. Per giustificare a destra una stringa in un campo viene utilizzato il simbolo di maggiore di (>).

Nella stringa di formato possono essere introdotti altri caratteri, che ventono trattati come caratteri alfabetici. Questo dà la possibilità di creare tabelle e diagrammi. Vedere la riga di programma 30 dell'esempio riportato di seguito.

ESEMPIO:

```
 5x=32;y=100.23:A$="GATTO":B$="COMPUTER"
 6F$="*#=########### * #$##.## *"+CHR$(13)*
10 PRINT USING "$###.##";13.25,x,y
20 PRINT USING "###>#";"CBM".A$
30 PRINT USING F $;A$,x,B$,y
```

CHR$(13)IS RETURN

All'esecuzione, la riga 10 stamperà:

$13.25 $32.00 $*****

Cinque asterischi (*****) vengono stampati al posto di un valore Y in quanto Y ha cinque cifre che non corrispondono alla lista di formato (come spiegato in precedenza).

La riga 20 stamperà:

CBM GATTO

Con due spazi prima di stampare la stringa, come definito nella lista di formato.

La riga 30 stamperà:

```
*    GATTO       *       $23.00*
*    COMPUTER    *      $100.23*
```

PUDEF

– Ridefinisce i simboli nell'istruzione PRINT USING

PUDEF "nnnn"

"nnnn" è una combinazione di caratteri, fino ad un massimo di quattro, e PUDEF permette di ridefinire ciascuno dei seguenti quattro simboli nell'istruzione PRINT USING: spazi, virgole, punti decimali e simboli di dollaro. Questi quattro simboli possono essere trasformati in altri caratteri introducendo il nuovo carattere alla posizione corretta nella stringa di controllo PUDEF.

La posizione 1 è il carattere riempitore. Il default è uno spazio vuoto. Introdurre in questo punto un nuovo carattere per sostituire il carattere di spazio vuoto con il carattere desiderato.

La posizione 2 è il carattere di virgola. Il default è una virgola.

La posizione 3 è il punto decimale. Il default è un punto decimale.

La posizione 4 è il simbolo di dollaro. Il default è un simbolo di dollaro.

ESEMPI:

10 PUDEF "*"

Stampa * al posto degli spazi vuoti.

20 PUDEF "<"

Stampa < al posto delle virgole.

NOTA: Devono essere specificate tutte le posizioni fino a quella da modificare.

Ad esempio PUDEF" $" stamperà $ nella posizione del simbolo di dollaro, mentre il punto decimale, la virgola ed il carattere riempitore saranno trasformati in spazi.
PUDEF verrà applicato solo ai formati numerici, cioè PUDERF"0" trasformerà gli spazi riempitori espressi in numeri in 0 iniziali ma non modificherà gli spazi riempitori espressi in stringhe.
Il carattere che sostituisce il simbolo $ non avrà effetto a meno che non sia preceduto da un simbolo # (mobile).

READ

– Legge i dati dalle istruzioni DATA e li introduce nella memoria del computer (mentre il programma è in fase di esecuzione)

READ lista variabili

Questa istruzione preleva le informazioni dalle istruzioni DATA e le memorizza in variabili, dove i dati possono essere utilizzati dal programma in corso di esecuzione. La lista di variabili dell'istruzione READ può contenere sia stringhe sia numeri. Occorre evitare di leggere stringhe quando l'istruzione READ attende un numero e viceversa, altrimenti verrà visualizzato il messaggio TYPE MISMATCH ERROR.

I dati nelle istruzioni READ vengono letti in ordine sequenziale. Ogni istruzione READ può leggere uno o più elementi di dati. Ogni variabile nell'istruzione READ richiede un elemento di dati. Se non viene fornito questo elemento, verrà visualizzato il messaggio OUT OF DATA ERROR.

All'interno di un programma è possibile leggere i dati e quindi leggerli nuovamente utilizzando l'istruzione RESTORE. RESTORE imposta il puntatore dei dati sequenziali all'inizio, in modo che i dati possano essere letti di nuovo. Vedere l'istruzione RESTORE.

ESEMPI:

10 READ A,B,C **20 DATA 3,4,5**	Legge 3 elementi di dati (che devono essere numerici per non causare un messaggio di errore) nelle variabili A,B e C.
10 READ A$,B$,C$ **20 DATA MARIO, PAOLO, GIORGIO**	Legge le tre stringhe dalle istruzioni DATA.
10 READ A,B$,C **20 DATA 1200,MARIA,345**	Legge una variabile numerica, una variabile stringa ed un'altra variabile numerica.

RECORD

– Posiziona i puntatori di file relativi

RECORD# numero file logico, numero record [,byte]

Questa istruzione posiziona un puntatore di file relativo per la selezione di tutti i byte (caratteri) di tutti i record in un file relativo. Il numero di file logico è compreso nella gamma da 1 a 255. Il numero di record può essere compreso nella gamma 1 a 65535. Il numero byte è compreso nella gamma da 1 a 254. Per dettagli sui file relativi consultare il manuale del disk drive.

Quando il numero di record viene impostato ad un valore maggiore dell'ultimo numero di record del file, avviene quanto segue:

In caso di un'operazione di scrittura (PRINT#), vengono creati record supplementari per espandere il file fino a raggiungere il numero di record desiderato.

In caso di un'operazione di lettura (INPUT#), viene ritornato un record nullo e viene visualizzato il messaggio "RECORD NOT PRESENT ERROR".

ESEMPI:

```
10 DOPEN #2"CLIENTI"
20 RECORD#2,10,1
30 PRINT#2,A$
40 DCLOSE #2
```

Questo esempio apre un file relativo esistente chiamato "CLIENTI" come numero di file 2 nella riga 10. La riga 20 posiziona il puntatore del file relativo sul primo byte nel record numero 10. La riga 30 scrive i dati, A$, sul file.

Il comando RECORD accetta le variabili come parametri. Si consiglia di introdurre un comando RECORD all'interno di un loop FOR...NEXT o DO. Vedere anche DOPEN.

REM

– Fornisce commenti o note per quanto riguarda lo svolgimento di una riga di programma

REM [messaggio]

L'istruzione REM rappresenta una nota per l'utente che sta leggendo un listato di programma. REM può dare spiegazioni su una parte di programma, sull'autore del programma stesso, ecc. Le istruzioni REM non alterano il funzionamento del programma, ma provocano solo un allungamento di esso (e quindi utilizzano dello spazio in memoria). Il computer non interpreta come istruzione eseguibile alcun dato a destra della parola chiave REM. Quindi sulla riga in cui si trova REM non potranno essere presenti istruzioni da eseguire.

ESEMPIO:

```
1010 NEXT X:REM Fine del loop del programma principale.
```

RENAME

– Modifica il nome di un file su disco

RENAME[Dnumero drive,]"vecchio nomefile" TO "nuovo nomefile" [<ON|,>Unumero dispositivo]

Questo comando viene utilizzato per assegnare un nuovo nome ad un file su disco. Il disk drive non rinomina un file se il file è aperto.

ESEMPI:

RENAME D0,"PROVA" TO "PRVFINALE"

Modifica il nome del file "PROVA" in "PRVFINALE".

RENAME D0,(A$) to (B$),U9

Modifica il nome di file specificato in A$ nel nome di file specificato in B$ nel drive 0, dispositivo 9. Quando viene utilizzato un nome di variabile come nome di file, ricordare di racchiuderlo tra parentesi.

RENUMBER

– Rinumera le righe di un programma BASIC

RENUMBER [nuovo numero riga iniziale][,incremento][,vecchio numero riga iniziale]

Nuova riga iniziale è il numero della prima riga del programma dopo la rinumerazione; il valore di default è 10.

Incremento è l'intervallo tra i numeri di riga (cioè10, 20, 30, ecc.); il valore di default per l'incremento è 10. Vecchio numero riga iniziale è il numero della prima riga prima della rinumerazione del programma. Questo permette la rinumerazione di una parte del programma. In questo caso il valore di default è la prima riga del programma. Questo comando può essere eseguito solo in modalità diretta.

Se il computer incontra un riferimento ad un numero di riga inesistente emetterà il messaggio "UNRESOLVED REFERENCE ERROR". Se la rinumerazione espande il programma oltre i limiti ammessi, verrà visualizzato il messaggio "OUT OF MEMORY". Verrà visualizzato il messaggio "LINE NUMBER TOO LARGE ERROR" se il comando RENUMBER genererà una riga con il numero 64000 o superiore. Entrambi questi messaggi non influiscono sul programma.

ESEMPI:

RENUMBER — Rinumera le righe del programma a partire da 10, con incrementi di 10.

RENUMBER 20,20,15 — Rinumera il programma partendo dalla riga 15. La riga 15 diventa la riga 20, e tutte le righe successive verranno incrementate di 20 in 20.

RENUMBER,,65 — Rinumera con incrementi di 10 partendo dalla riga 65. La riga 65 diventa la riga 10. Se si desidera omettere un parametro, al suo posto si dovrà inserire una virgola. Non ci devono essere righe tra la 10 e la 64 compresa.

Salvare sempre i programmi prima di effettuare la rinumerazione in quanto programmi molto lunghi possono causare un crash di sistema quando vengono rinumerati con numeri di riga elevati.

Notare inoltre che i programmi lunghi devono essere rinumerati nella modalità veloce in quanto la numerazione viene effettuata in un tempo piuttosto lungo (fino a 30 minuti per un programma da 55K nella modalità veloce).

Se si utilizza la modalità a 40 colonne, battere FAST:RENUMBER...<RETURN> quindi battere SLOW<RETURN>. Durante l'operazione di rinumerazione, sullo schermo non apparirà nulla fino al termine dell'operazione stessa.

Se si utilizza la visualizzazione a 80 colonne, selezionare lo schermo a 80 colonne prima di battere FAST.

RESTORE

– Reimposta il puntatore READ in modo che i dati possono essere letti nuovamente

RESTORE **Modalità C64**

RESTORE [riga #] **Modalità C128**

Quando viene eseguito all'interno di un programma, il puntatore di un elemento in un'istruzione DATA che dovrà essere letto, viene reimpostato sul primo elemento dell'istruzione DATA. Questo dà la possibilità di leggere nuovamente i dati. Se l'istruzione RESTORE è seguita da un numero di riga, il puntatore READ viene impostato sul primo elemento dati dopo la riga di programma specificata. In caso contrario il puntatore verrà reimpostato all'inizio del programma BASIC.

Nella modalità C64 non esiste un'opzione per specificare il numero di riga, quindi si dovrà utilizzare RESTORE a partire dall'inizio del programma.

ESEMPI:

```
10 FOR I = 1 TO 3
20 READ X
30 T = X + T
40 NEXT
45 PRINT T
50 RESTORE
60 GOTO 10
70 DATA 10.20,30
```

Questo esempio legge i dati nella riga 70 e li memorizza nella variabile numerica X. Aggiunge il totale di tutti gli elementi di dati numerici. Dopo la lettura di tutti i dati per tre volte, il puntatore READ viene portato all'inizio del programma alla riga 10 e continuerà l'esecuzione all'infinito.

```
10 READ A,B,C
20 DATA 100,500,750
30 READ X,Y,Z
40 DATA 36,24,38
50 RESTORE 40
60 READ S,P,Q
70 PRINT A,B,C
80 PRINT X,Y,Z
90 PRINT S,P,Q
```

Questo esempio riporta il puntatore DATA all'inizio dell'elemento dati nella riga 40. Dopo l'esecuzione della riga 60, il puntatore leggerà i dati 36,24,38 sulla riga 40, in quanto non è più necessario leggere nuovamente i dati della riga 20.

NOTA: Quando si specifica un numero di riga, accertarsi che la riga esista effettivamente. È impossibile utilizzare una variabile, ad esempio **RESTORE LR**.

RESUME

– Definisce il punto da cui riprenderà l'esecuzione del programma dopo l'individuazione di un errore

RESUME [riga #|NEXT]

Questa istruzione viene utilizzata per riavviare l'esecuzione del programma dopo l'individuazione di un errore per mezzo di TRAP. Se non vengono introdotti parametri, RESUME tenterà di rieseguire la riga in cui si è verificato l'errore. RESUME NEXT riprende l'esecuzione dall'istruzione immediatamente successiva a quella contenente l'errore; RESUME seguito da un numero di riga si sposterà sulla riga specificata e riprenderà l'esecuzione da quella riga. RESUME può essere utilizzato solo in modalità programma.

ESEMPIO:

```
 10 TRAP 100
 20 INPUT "INTRODUCI UN NUMERO",A
 30 B=100/A
 40 PRINT "IL RISULTATO =",B:PRINT"FINE"
 50 PRINT "VUOI ESEGUIRLO NUOVAMENTE (S/N)"
    :GETKEY Z$:IF Z$="S" THEN 20
 60 STOP
100 INPUT "INTRODUCI UN ALTRO NUMERO
    (NON ZERO)";A
110 RESUME
```

Questo esempio individua un errore di "divisione per zero" nella riga 30 se viene introdotto 0 nella riga 20. Se si introduce zero, il programma salta alla riga 100, dove si chiede di introdurre un numero che non sia zero. La riga 110 ritorna alla riga 30 per completare il calcolo. La riga 50 chiede se si desidera ripetere nuovamente il programma. In caso affermativo, premere il tasto S.

RETURN

– Esce dalla subroutine

RETURN

Questa istruzione viene sempre utilizzata con l'istruzione GOSUB. Quando il programma incontra un'istruzione RETURN, va all'istruzione immediatamente successiva all'ultimo comando GOSUB eseguito. In caso non fosse stato emesso precedentemente un comando GOSUB, verrà visualizzato il messaggio RETURN WITHOUT GOSUB ERROR ed il programma si arresterà.
Tutte le subroutine terminano con un'istruzione RETURN.

ESEMPIO:

```
 10 PRINT "INIZIO SUBROUTINE"
 20 GOSUB 100
 30 PRINT "FINE SUBROUTINE"
  .
  .
  .
 90 STOP
100 PRINT "SUBROUTINE 1"
110 RETURN
```

Questo esempio chiama la subroutine alla riga 100 che stampa il messaggio "SUBROUTINE 1" e ritorna alla riga 30, la parte rimanente del programma.

RUN

– Esegue un programma BASIC

1) **RUN [riga#]**
2) **RUN "nomefile" [,Dnumero drive][,Unumero dispositivo]**

Solo in modalità C128

Dopo l'introduzione o caricamento di un programma in memoria, il comando RUN lo esegue. Prima di iniziare l'esecuzione del programma RUN cancella tutte le variabili nel programma stesso. Se il comando RUN è seguito da un numero, l'esecuzione inizierà da quel numero di riga. Se il comando RUN è seguito da un nome di file, questo file verrà caricato dal disk drive ed eseguito, senza altre azioni da parte dell'utente. RUN può essere utilizzato all'interno di un programma. Il numero di default per il drive è 0, mentre per il dispositivo è 8.

ESEMPI:

RUN	Avvia l'esecuzione a partire dall'inizio del programma al momento in memoria.
RUN 100	Avvia l'esecuzione del programma a partire dalla riga 100.
RUN"PRG1"	Carica (DLOAD) "PRG1" dal disk drive 8 ed esegue questo programma a partire dalla prima riga.
RUN(A$)	Carica (DLOAD) il programma denominato nella variabile A$ e lo esegue a partire dalla prima riga.

SAVE

– Memorizza su disco o nastro il programma in memoria

SAVE ["nomefile"][,numero dispositivo][,flag EOT]

Questo comando memorizza su nastro o disco il programma correntemente in memoria.

Se SAVE viene battuto da solo, sul nastro verrà salvato un file senza nome. Il nastro è un sistema sequenziale e l'utente dovrà accertarsi prima del salvataggio che sul nastro non siano presenti informazioni che potrebbero essere ancora necessarie (vedere VERIFY). Per assegnare un nome al programma, racchiudere il nome desiderato tra virgolette (oppure utilizzare una variabile stringa) subito prima di battere SAVE. Il nome dovrà essere composto da un massimo di 16 caratteri.

NOTA: In fase di salvataggio su disco, si dovrà specificare un nome di file, altrimenti verrà visualizzato il messaggio "MISSING FILE NAME ERROR".

Per specificare il numero di dispositivo (es. 1 per il nastro) battere una virgola seguita dal numero di dispositivo dopo le virgolette di chiusura del nome di file.

Il parametro finale (EOT) segue il numero di dispositivo e anche questo è separato da una virgola. Se si utilizza un disco questo parametro non ha alcun significato, mentre se si utilizza un nastro potrà avere uno dei seguenti quattro valori:

0 Default - Nessuna azione

1 Esegue il salvataggio in modo che la funzione di riallocazione di LOAD non abbia alcun effetto (cioè il file verrà caricato sempre a partire dall'indirizzo da cui era stato salvato).

2 Introduce un'indicazione di FINE NASTRO (EOT) alla fine del file - Se si tenta di caricare informazioni oltre la fine del file viene visualizzato il messaggio "FILE NOT FOUND ERROR".

3 Salva in formato non riallocabile (1) ed introduce un EOT (2).

NOTA: Se si specifica il numero di dispositivo o il parametro EOT, si dovrà introdurre il nome del file (e il numero di dispositivo). Se si utilizza il nastro, è possibile introdurre un carattere nullo ("   "). Vedere gli esempi seguenti.

ESEMPI:

SAVE "CIAO"	Memorizza un programma su nastro con il nome CIAO.
SAVE A$,8	Memorizza un programma su disco, con il nome memorizzato nella variabile A$.
SAVE "CIAO",8	Memorizza su disco, con il nome CIAO (operazione equivalente a DSAVE "CIAO").
SAVE "CIAO",1,2	Memorizza su nastro, con il nome CIAO ed introduce un indicatore di FINE NASTRO dopo il programma.
SAVE" ",1,3	Memorizza su nastro, senza assegnare un nome, introduce un indicatore di FINE NASTRO (EOT) alla fine del programma e non permette la riallocazione o il caricamento del programma.

SCALE

– Modifica la scala in modalità grafica

SCALE n [,xmax,ymax]

dove:

n = 1 (attivato), oppure 0 (disattivato)

xmax è un valore compreso nella gamma da 320 a 32767, default 1023 (alta ris.) 2047 (multicolore).
ymax è un valore compreso nella gamma da 200 a 32767, default 1023.

Modifica la scala delle coordinate di visualizzazione a matrice di punti nelle modalità multicolore e ad alta risoluzione. Vengono scalate anche le coordinate del comando MOVSPR. Effettua il mappaggio di molti punti logici in un solo punto fisico.

Questa funzione è utile quando si desidera tracciare dati su una vasta gamma di valori, ma non se si possiede un alto numero di dati con valori elevati.

La modalità multicolore utilizza 2 pixel fisici per punto sull'asse x, quindi la visualizzazione normale sarà

X = da 0 a 159; Y = da 0 a 199

in opposizione a

X = da 0 a 319; Y = da 0 a 199

Se si desiderano utilizzare le stesse coordinate per multicolore e alta risoluzione, utilizzare SCALE 1,640,200 dopo l'impostazione di uno schermo multicolore ed utilizzare i valori di default SCALE per i due tipi di schermo.
NOTA: Il comando GRAPHIC disattiva la scala, è quindi equivalente a GRAPHIC...:SCALE 0.

ESEMPIO:

```
 10 GRAPHIC 1:GOSUB 100
 20 SCALE 1:GOSUB 100
 30 SCALE 1,5000,5000:GOSUB 100
 40 END
100 CIRCLE 1,160,100,60:RETURN
```

SCNCLR

– Cancella lo schermo

SCNCLR [numero modalità]

Le modalità sono le seguenti:

Numero modalità	Modalità
0	testo a 40 colonne (VIC)
1	matrice di punti*
2	matrice di punti a schermo diviso*
3	matrice di punti multicolore*
4	matrice di punti multicolore a schermo diviso*
5	testo a 80 colonne (8563)

Questa istruzione non seguita da alcun argomento, cancellerà lo schermo grafico, in caso contrario cancellerà lo schermo di testo corrente. **

ESEMPI:

SCNCLR 5 Cancella lo schermo di testo a 80 colonne.

SCNCLR 1 Cancella lo schermo a matrice di punti (VIC).

SCNCLR 4 Cancella lo schermo a matrice di punti multicolore a schermo diviso (VIC).

NOTA:* L'area a matrice di punti è uguale per alta risoluzione e multicolore, i diversi numeri di modalità selezionano altri parametri per cancellare ad esempio la ram di colore (3 e 4) del testo a 40 colonne (2 e 4).

** Se è stato creato uno schermo grafico ma non è stato selezionato (GRAGPHIC=0), questo sarà cancellato. Se si stanno utilizzando due schermi (a 80 colonne per testo e 40 colonne per grafica), SCNCLR cancellerà sia lo schermo grafico sia quello di testo se richiamato dallo schermo a 80 colonne.

SCRATCH

– Cancella un file dall'elenco del disco

SCRATCH "nomefile" [,Dnumero drive][<ON|,>[,Unumero dispositivo]

Questo comando cancella un file dall'elenco del disco. Per precauzione il computer chiede conferma visualizzando "ARE YOU SURE?" (solo per la modalità diretta) prima del completamento dell'operazine. Battere Y per eseguire l'istruzione SCRATCH, oppure un qualsiasi altro tasto per annullare l'operazione. Utilizzare questo comando per cancellare file inutili e per liberare spazio sul disco. Il nome di file può contenere un carattere variabile (?,* ecc.). Il numero di drive di default è 0 ed il numero di dispositivo di default è 8.

ESEMPIO:

SCRATCH "MIOFILE",D0

Questo cancellerà il file MIOFILE dal disco nel drive 0, dispositivo 8.

SLEEP

– Ritarda il programma per un periodo di tempo specificato

SLEEP N

dove N rappresenta i secondi 0< N<65535

Per disattivare un ritardo troppo lungo per un programma, premere il tasto STOP.

SLOW

– Riporta il Commodore 128 al funzionamento normale a 1 MHz

SLOW

Il Commodore 128 può far funzionare il microprocessore 8502 ad una velocità di 1 o 2 MHz.

Il comando SLOW rallenta il microprocessore da 2 MHz a 1 MHz. Il comando FAST imposta il Commodore 128 sul funzionamento a 2 MHz. Il Commodore 128 può elaborare i comandi in modo molto più veloce a 2 MHz di quanto non possa fare a 1 MHz.

In ogni caso lo schermo a 40 colonne non può essere utilizzato a 2 MHz.

SOUND

– Produce effetti sonori e note musicali

SOUNDv,f,d[,dir][,m][,g][,o][,e]

dove:

- **v** = voce (1..3)
- **f** = valore di frequenza (0..65535)
- **d** = durata (0..32767)
- **dir** = direzione di incremento (0(verso l'alto),1(verso il basso) oppure 2(oscillazione)) default = 0
- **m** = frequenza minima (se viene utilizzato il glissato) (0..65535) default = 0
- **g** = valore di incremento per il glissato (0..65535) default = 0
- **o** = forma d'onda (0 = triangolo, 1 = dente di sega, 2 = impulso, 3 = rumore) default = 2
- **l** = larghezza impulso (0..4095) default = 2048

Il comando SOUND permette di creare velocemente e facilmente effetti sonori e toni musicali. I tre parametri richiesti v, f e d selezionando la voce, la frequenza e la durata del suono. La durata è in unità chiamate jiffy. 60 jiffy equivalgono a un secondo.

Il comando SOUND permette di glissare da una frequenza all'altra producendo effetti sonori su una vasta gamma di note. Specificare la direzione del glissato con il parametro DIR. Impostare la frequenza minima del glissato con M, e il valore di incremento del glissato con G. Selezionare l'appropriata forma d'onda con O, e specificare con L la larghezza della forma d'onda ad un impulso variabile se questa è stata selezionata in O.

ESEMPI:

SOUND 1,40960,60 — Emette un suono alla frequenza 40960 nella voce 1 per un secondo.

SOUND 2,2000,50,0,2000,100 — Emette un suono glissato dalla frequenza 2000 e incrementando verso l'alto con unità di 100.

SOUND 3,5000,1,2, 3000,500,1 — Questo esempio emette una gamma di suoni a partire dalla frequenza minima di 3000 fino a 5000 con incrementi di 500. La dire-

zione del glissato è in avanti e all'indietro (oscillante). La forma d'onda selezionata è il dente di sega e la voce selezionata è la 3.

SPRCOLOR

– Imposta il multicolor 1 e/o multicolor 2 per tutti gli sprite (solo per visualizzazione a 40 colonne).

SPRCOLOR[smrc-1][,smcr-2]

dove:

smcr-1	Imposta multicolor 1 per tutti gli sprite
smcr-2	Imposta multicolor 2 per tutti gli sprite

Ognuno di questi parametri può essere qualsiasi colore da 1 a 16.

ESEMPI:

SPRCOLOR 3,7	Imposta il multicolor 1 dello sprite su rosso e multicolor 2 su blu.
SPRCOLOR 1,2	Imposta il multicolor 1 dello sprite su nero e multicolor 2 su bianco.

SPRDEF

– Attiva la modalità di definizione sprite per creare e modificare immagini di sprite (solo per visualizzazione a 40 colonne).

SPRDEF

Il comando SPRDEF definisce gli sprite interattivamente.

Il comando SPRDEF provocherà la visualizzazione di un'area di lavoro di sprite sullo schermo (24 caratteri di larghezza per 21 di altezza). Ogni posizione di carattere nella griglia corrisponde ad un pixel di sprite nello sprite visualizzato a destra dell'area di lavoro. Di seguito vengono elencate le operazioni della modalità di definizione sprite ed i tasti relativi.

Introduzione dell'utente	Descrizione
tasto RETURN	Esce dalla modalità di progettazione sprite al prompt SPRITE NUMBER?
1–8	Seleziona un numero di sprite.
A	Attiva e disattiva lo spostamento automatico del cursore.
tasti CRSR	Spostano il cursore.
tasto RETURN	Sposta il cursore all'inizio della nuova riga.
tasto HOME	Sposta il cursore all'angolo superiore sinistro dell'area di lavoro di sprite.
tasto CLR	Cancella l'intera griglia.
1–4	Seleziona la sorgente di colore. 1 cancellazione 2 primo piano 3 multicolor 1 4 multicolor 2
tasto <CTRL 1-8>	Seleziona il colore di primo piano dello sprite (1-8).
tasto Commodore,1-8	Seleziona il colore di primo piano dello sprite (9-16).
tasto STOP	Annulla le modifiche e ritorna al prompt.
SHIFT RETURN	Memorizza lo sprite e ritorna al prompt SPRITE NUMBER?
X	Espande lo sprite nella direzione X (orizzontale).
Y	Espande lo sprite nella direzione Y (verticale).
M	Sprite multicolor.
C	Copia i dati di sprite da uno sprite all'altro.

NOTA: Se si utilizza SPRDEF si causerà la cancellazione dello schermo a matrice di punti.

SPRITE

– Attiva e disattiva, colora, espande e imposta le priorità di schermo per uno sprite

SPRITE <numero>[,1/0][,prpi][,priorità][,x-esp][,y-esp] [,modalità]

L'istruzione SPRITE controlla la maggior parte delle caratteristiche di uno sprite.

Parametro	Descrizione
numero	Numero di sprite (1-8)
1/0	Attiva (1) lo sprite, o disattiva (0) lo sprite.
prpi	Colore di primo piano dello sprite (1-16)
priorità	Priorità è 0 se gli sprite devono apparire davanti agli oggetti sullo schermo. Priorità è 1 se gli sprite devono apparire dietro gli oggetti sullo schermo.
x-esp	Espansione orizzontale attivata (1) o disattivata (0).
y-esp	Espansione verticale attivata (1) o disattivata (0).
modalità	Seleziona lo sprite standard (0) o lo sprite multicolor (1).

I parametri non specificati in istruzioni di SPRITE successive assumono le caratteristiche dell'istruzione sprite precedente. È possibile controllare le caratteristiche di uno sprite con la funzione RSPRITE.

ESEMPI:

SPRITE 1,1,3 — Attiva lo sprite numero 1 e lo colora di rosso.

SPRITE 2,1,7,1,1,1 — Attiva lo sprite numero 2 e lo colora di blu. Fa passare lo sprite dietro agli oggetti sullo schermo e lo espande sia in orizzontale sia in verticale.

SPRITE 6,1,1,0,0,1,1 — Attiva lo sprite numero 6 e lo colora di nero. Il primo 0 indica al computer di visualizzazione gli sprite davanti agli oggetti sullo schermo. Il secondo 0 e l'1 successivo indicano al C128 di espandere lo sprite solo verticalmente. L'ultimo 1 specifica la modalità multicolor. Utilizzare il comando SPRCOLOR per selezionare il multicolor dello sprite.

SPRITE 7,,,,1 — Imposta l'espansione orizzontale dello sprite 7; tutte le altre opzioni rimangono invariate.

SPRSAV

– Memorizza i dati di sprite da una variabile di stringa di testo a un'area di memorizzazione sprite o viceversa

SPRSAV origine,destinazione

Questo comando trasferisce un'immagine di sprite da una variabile di stringa a un'area di memorizzazione sprite. Inoltre può trasferire i dati dall'area di memorizzazione sprite ad una variabile di stringa. Sia l'origine sia la destinazione possono essere un numero di sprite o una variabile di stringa ma non possono essere entrambe variabili di stringa. Se si sta spostando una stringa in uno sprite, solo i primi 63 byte di dati vengono utilizzati. I rimanenti byte vengono ignorati dal momento che uno sprite può contenere solamente 63 byte di dati.

ESEMPI:

SPRSAV 1,A$	Trasferisce la configurazione dell'immagine dallo sprite 1 alla stringa definita in A$.
SPRSAV B$,2	Trasferisce i dati dalla variabile di stringa B$ allo sprite 2.
SPRSAV 2,3	Trasferisce i dati dallo sprite 2 allo sprite 3.

NOTA: La stringa SPRSAV sprite, produce una stringa nello stesso formato di SSHAPE in mod che possa essere utilizzata con GSHAPE per 'fissare' uno sprite sullo schermo ad alta risoluzione. La stringa avrà una lunghezza di 67 caratteri.

SSHAPE/GSHAPE

– Salva/richiama figure da/a variabili di stringa

SSHAPE e GSHAPE vengono utilizzate per salvare e caricare aree rettangolari di schermi multicolor o a matrice di punti da/a variabili di stringa BASIC. Il comando per salvare un'area di schermo in una variabile di stringa è:

SSHAPE variabile di stringa,X1,Y1[,X2,Y2]

dove:

variabile di stringa	Nome della stringa in cui vengono salvati i dati
X1,Y1	Coordinata d'angolo (da 0,0 a 320,200) (scalata)
X2,Y2	Coordinata d'angolo opposto (X1,Y1) (il default è CP)

Dato che il BASIC limita la grandezza delle stringhe a 255 caratteri, la dimensione dell'area che può essere salvata è limitata. La dimensione di stringa richiesta può essere calcolata utilizzando una delle seguenti formule (non scalate):

L(h–r)=INT((ABS(x1–x2)+1)/8+.99)*(ABS(y1–y2)+1)+4

L(mcm)=INT((ABS(x1–x2)+1)/4+.99)*(ABS(y1–y2)+1)+4

NOTA: Se x2-x1 è 23 e la modalità grafica è alta risoluzione (modalità 1 o 2), una stringa prodotta con SSHAPE può essere utilizzata per generare uno sprite (vedere SPRSAV).

Il comando per richiamare (caricare) i dati da una variabile di

stringa e per visualizzarli su coordinate specifiche di schermo è:

GSHAPE variabile di stringa[X,Y][,modalità,]

dove:

stringa	Contiene la forma da tracciare.
X,Y	Coordinata superiore sinistra (da 0,0 a 319,199) che indica dove tracciare la forma (scalata - il default è il cursore pixel).
modalità	Modalità di sostituzione: 0: posiziona la forma così come è (default). 1: inverte la forma. 2: esegue sulla forma un OR con l'area. 3: esegue sulla forma un AND con l'area. 4: esegue sulla forma un XOR con l'area.

La modalità di sostituzione permette di modificare i dati nella variabile di stringa: invertire la forma, eseguire un OR logico, un OR esclusivo, o un AND esclusivo sull'immagine. Vedere anche il comando LOCATE per ulteriori informazioni sul cursore pixel.

ESEMPI:

SSHAPE A$,10,10	Salva un'area rettangolare dalle coordinate 10,10 alla locazione del cursore pixel, nella variabile di stringa A$.
SSHAPE B$,20,30,47,51	Salva un'area rettangolare dalla coordinata superiore sinistra (20,30) alla coordinata inferiore destra (47,51) nella variabile di stringa B$.
GSHAPE A$,120,20	Richiama la forma contenuta nella variabile di stringa A$ e la visualizza alla coordinata superiore sinistra (120,20).

GSHAPE B$,30,30,1	Richiama la forma contenuta nella variabile di stringa B$ e la visualizza alla coordinata superiore sinistra 30,30. La forma viene invertita avendo introdotto la modalità di sostituzione 1.

NOTA: Fare attenzione nell'utilizzo delle modalità 1–4 con forme multicolori. Si potrebbero ottenere risultati imprevisti.

STASH

– Trasferisce il contenuto della memoria host alla RAM di espansione

STASH#byte,iniz,inesp,nbes

Consultare il comando FETCH per la descrizione dei parametri.

STOP

– Interrompere l'esecuzione del programma

STOP

Questa istruzione interrompe il programma. Verrà visualizzato il messaggio BREAK IN LINE XXX (solo in modalità programma) dove XXX è il numero di riga contenente il comando STOP. Il programma può essere ripreso dall'istruzione successiva a STOP utilizzando il comando CONT se non sono state apportate delle modifiche al listato. L'istruzione STOP viene spesso utilizzata durante la messa a punto del programma.

SWAP

– Scambia il contenuto della RAM host con il contenuto della RAM di espansione

SWAP#byte,iniz,inesp,nbes

Consultare il comando FETCH per la descrizione dei parametri.

SYS

– Richiama ed esegue una subroutine in linguaggio macchina all'indirizzo specificato

SYS indirizzo	Modalità C64
SYS indirizzo[,a][,x][,y][,s]	Modalità C128

Questa istruzione esegue il richiamo ad una subroutine in codice macchina ad un indirizzo dato in un'impostazione di configurazione di memoria in accordo con il comando BANK. Come opzioni gli argomenti di A,X,Y e S vengono caricati nei registri di accumulatore di X e Y e di stato rispettivamente prima del richiamo della subroutine. La gamma di indirizzo è da 0 a 65535. Il programma inizia l'esecuzione del programma in linguaggio macchina a partire da quella locazione di memoria. Vedere anche il comando BANK.

ESEMPI:

SYS 40960	Richiama ed esegue la routine in linguaggio macchina alla locazione 40960.
SYS 8192,0	Richiama ed esegue la routine in linguaggio macchina alla locazione 8192 e carica 0 nell'accumulatore.

TEMPO

– Definisce la velocità dell'emissione musicale

TEMPO n

dove n è la durata relativa (tra 1 e 255).

La durata effettiva di una nota intera viene determinata utilizzando la formula seguente:

durata della nota intera = 23,06/n secondi

Il valore di default è 8, e la durata della nota aumenta con n.

ESEMPI:

TEMPO 16	Definisce il tempo a 16.
TEMPO 1	Definisce il tempo alla velocità minima.
TEMPO 250	Definisce il tempo a 250.

TRAP

– Individua e gestisce errori di programma durante l'esecuzione di un programma BASIC.

TRAP[riga#]

Se attivato, TRAP individua la maggior parte delle condizioni di errore, (escluso messaggi di errore DOS, ma compreso il tasto STOP). Nel caso di un qualsiasi errore di esecuzione il flag di errore viene impostato e l'esecuzione viene trasferita al numero di riga specificato nell'istruzione TRAP. Il numero di riga in cui l'errore è stato individuato può essere ritrovato utilizzando la variabile di sistema EL. La condizione di errore specificata è contenuta nella variabile di sistema ER. La funzione di stringa ERR$(ER) fornisce il messaggio di errore corrispondente a una condizione di errore.

L'istruzione RESUME può essere utilizzata per riprendere l'esecuzione del programma. TRAP senza numero di riga disattiva l'individuazione degli errori. Un errore in una routine TRAP non può essere individuato.

ESEMPI:

100 TRAP 1000	Se viene incontrato un errore salta alla riga 1000.
1000?ERR$(ER);EL	Stampa un messaggio di errore e il numero di riga dell'errore.
1010 RESUME	Riprende l'esecuzione del programma.

TROFF

– Disattiva la modalità di tracciamento

TROFF

Quest'istruzione disattiva la modalità di tracciamento.

TRON

– Attiva la modalità di tracciamento

TRON

TRON viene utilizzato nella messa a punto dei programmi. Questa istruzine attiva la modalità di tracciamento. Quando si esegue il programma i numeri di riga del programma appaiono racchiusi tra parentesi prima dell'esecuzione dell'azione richiesta dalla riga stessa.

Se vi sono righe a più istruzioni, il numero di riga sarà stampato prima dell'elaborazione di ogni istruzione.

VERIFY

– Confronta il programma in memoria con quello salvato su disco o su nastro

VERIFY"nomefile"[,numero dispositivo][,flag di rilocazione]

Questo comando ordina al Commodore 128 di confrontare il programma su nastro o su disco con quello in memoria, per determinare se il programma in memoria è stato effettivamente salvato. Questo comando è inoltre molto utile per salvare un programma subito dopo l'ultimo programma presente su nastro.

VERIFY senza argomenti ordina al Commodore 128 di confrontare il primo programma trovato su nastro senza tener conto del suo nome, con il programma presente in memoria. VERIFY seguito da un nome di programma fra virgolette o da una variabile di stringa ricerca sul nastro il programma specificato e lo confronta con quello presente in memoria. VERIFY seguito da un nome, una virgola e un numero confronta il programma sul dispositivo corrispondente al numero indicato (1 per il nastro, 8 per il disco). Il flag di rilocazione è lo stesso del comando LOAD (verifica il programma dalla locazione di memoria da cui è stato salvato). Vedere anche DVERIFY.

ESEMPI:

VERIFY	Confronta il primo programma incontrato su nastro.

VERIFY"CIAO"	Ricerca il programma CIAO su nastro e lo confronta con quello presente in memoria.
VERIFY"CIAO",8,1	Ricerca il programma CIAO su disco e lo confronta con quello presente in memoria.
VERIFY"ultimo file"	Ricerca sul nastro l'ultimo file, controlla e visualizza un messaggio di errore se non esiste corrispondenza. A questo punto è possibile salvare il nuovo programma dopo questo file senza cancellare i programmi precedenti.

NOTA: Se viene riallocata un'area grafica da utilizzarsi dopo un'operazione di salvataggio, VERIFY e DVERIFY provocheranno un messaggio di errore. Tecnicamente questa operazione è corretta. Il testo BASIC in questo caso è stato spostato dalla sua locazione originale ad un'altra gamma di indirizzi. Per cui VERIFY, che esegue un confronto byte/byte, darà esito negativo anche se il programma è valido.

VOL

– Definisce il livello sonoro di uscita

VOL livello sonoro

Questa istruzione imposta il volume per le istruzioni SOUND e PLAY. Il volume può essere impostato da 0 a 15, dove 15 è il volume massimo e 0 è quello minimo. VOL ha effetto su tutte le voci.

ESEMPI:

VOL 0	Abbassa completamente il volume.
VOL 1	Imposta il volume al valore minimo.
VOL 15	Imposta il volume per le istruzioni SOUND e PLAY al valore massimo.

WAIT

– Interrompere l'esecuzione del programma fino alla verifica di una condizione

WAIT Locazione,maschera–1,[,maschera–2]

L'istruzione WAIT provoca l'interruzione momentanea dell'esecuzione del programma fino al riconoscimento da parte dell'indirizzo di memoria dato della configurazione di bit specificata o di un valore. In altre parole WAIT può essere utilizzata per interrompere un programma fino all'intervento di evento esterno. Questa operazione può essere effettuata controllando lo stato dei bit nei registri di input/output. Gli elementi dei dati utilizzati con l'istruzione WAIT possono avere un valore qualsiasi. Per la maggior parte dei programmatori quest'istruzione risulterà inutile, in quanto causa l'arresto del programma fino alla modifica particolare dei bit di una locazione di memoria specifica. Questo tipo di funzione viene utilizzato solo per particolari operazioni di I/O. L'istruzione WAIT prende il valore nella locazione di memoria ed esegue un AND logico con il valore in maschera 1. Se viene specificata "maschera 2", sul risultato della prima operazione viene effettuato un OR esclusivo con maschera 2. In altre parole maschera 1 esclude i bit che non sono da confrontare. Dove il bit è 0 in maschera 1, il bit corrisponde nel risultato sarà sempre 0. Il valore in maschera 2 commuta tutti i bit in modo da poter controllare sia la condizione ON sia la condizione OFF. Ogni bit controllato come 0 deve avere un 1 nella posizione corrispondente in maschera 2. Se i bit corrispondenti degli operandi in maschera 1 e maschera 2 differiscono, l'operazione di OR esclusivo dà un risultato di 1. Se i bit corrispondenti ottengono lo stesso risultato il bit sarà 0. È possibile introdurre una pausa infinita con l'istruzione WAIT, in tal caso i tasti RUN/STOP e RESTORE possono essere utilizzati per riprendere l'esecuzione. WAIT può richiedere un comando BANK se la memoria a cui si desidera accedere non è nel blocco correntemente selezionato.

Gli esempi seguenti sono applicabili solo con la modalità C128. Il primo esempio attende la pressione di un tasto per poter continuare il programma. Il secondo esempio attende la pressione e quindi il rilascio del tasto SHIFT. Il terzo esempio attende finchè il bit 7 (128) è attivato o il bit 4 (16) è disattivato.

ESEMPI:

WAIT 1,32,32
WAIT 211,1:WAIT 211,1,1
WAIT 36868,144,16

(144 e 16 sono maschere binarie. 144 = 10010000 in binario e 16=10000 in binario).

WIDTH

– Imposta l'ampiezza delle righe tracciate.

WIDTH n

Questo comando imposta l'ampiezza delle righe tracciate utilizzando i comandi grafici BASIC su singola o doppia ampiezza. Dando a n un valore di 1, viene impostata la singola ampiezza; il valore di 2 imposta la doppia ampiezza.

ESEMPI:

WIDTH 1	Imposta l'ampiezza singola per i comandi grafici.
WIDTH 2	Imposta l'ampiezza doppia per le righe tracciate.

WINDOW

– Definisce una finestra di schermo

WINDOW col sup sin, rig sup sin, col inf des, ring inf des [,cancellazione]

Questo comando definisce una finestra logica all'interno dello schermo di testo a 40 o a 80 colonne. Le coordinate devono essere nella gamma 0-39/79 per i valori di colonna e 0-24 per i valori di riga. Il flag di cancellazione se indicato (1) provoca la cancellazione dello schermo (solo all'interno dei limiti della finestra corrente).

ESEMPI:

WINDOW 5,5,35,20

Definisce una finestra con coordinata dell'angolo superiore sinistro 5,5 e coordinata dell'angolo inferiore destro 35,20.

WINDOW 10,2,33,24,1

Definisce una finestra con coordinata dell'angolo superiore sinistro 10,2 e coordinata dell'angolo inferiore destro 33,24. Inoltre cancella la porzione di schermo contenuta dalla finestra come indicato dal parametro 1.

NOTA: Se si specifica una colonna maggiore di 39 in modalità a 40 colonne, verrà visualizzato il messaggio "ILLEGAL QUANTITY ERROR".

SEZIONE 18

Funzioni Basic

Il formato della descrizione della funzione è:

FUNZIONE (argomento)

dove argomento può essere un valore numerico, variabile o stringa.

Ogni descrizione di funzione è seguita da un ESEMPIO. Le righe che vengono evidenziate in grasetto negli esempi sono le funzioni introdotte dall'utente. La riga di caratteri normali è la risposta del computer.

ABS

– Ritorna un valore assoluto

ABS (X)

La funzione di valore assoluto ritorna il valore positivo dell'argomento.

ESEMPIO:

PRINT ABS (7*(–5))

35

ASC

– Ritorna il codice ASCII CBM del carattere

ASC(X$)

Questa funzione ritorna il codice ASCII del primo carattere di X$. Nella modalità C128 non sarà così più necessario aggiungere CHR$(0) ad una stringa nulla. Il messaggio ILLEGAL QUANTITY ERROR non sarà più visualizzato.

ESEMPIO:

X$="C128":PRINT ASC (X$)

67

ATN

– Ritorna l'angolo la cui tangente è X radianti

ATN (X)

Questa funzione ritorna l'angolo la cui tangente è X, misurata in radianti.

ESEMPIO:

PRINT ATN(3)

1.24904577

BUMP

– Ritorna l'informazione di collisione sprite

BUMP (N)

Questa funzione viene utilizzata per determinare quali sprite sono entrati in collisione dall'ultimo controllo effettuato. BUMP(1) registra quali sprite sono entrati in collisione con altri sprite mentre BUMP(2) registra quali sprite sono entrati in collisione con altri oggetti sullo schermo. Per utilizzare BUMP, COLLISION non deve essere attivo. Le posizioni dei bit (da 0 a 7) del valore BUMP corrispondono rispettivamente agli sprite da 1 a 8. BUMP(n) viene reimpostato a zero dopo ogni chiamata.

Il valore ritornato da BUMP è il risultato di due elevato alla potenza della posizione del bit. Ad esempio, se BUMP ritorna un valore 16, questo significa che lo sprite entrato in collisione è lo sprite 4, in quanto 2 alla quarta equivale a 16.

ESEMPI:

PRINT BUMP(1)
12

Indica che si sono scontrati gli sprite 2 e 3.

PRINT BUMP(2)
32

Indica che lo sprite 5 è entrato in collisione con un oggetto dello schermo.

CHR$

– Ritorna il carattere ASCII del codice ASCII CBM specificato

CHR$(X)

Questa funzione è l'opposto di ASC e ritorna il carattere stringa il cui codice ASCII CBM è X. Fare riferimento all'Appendice E per avere una tabella dei codici CHR$.

ESEMPI:

PRINT CHR$ (65) Stampa il carattere a.
a
PRINT CHR$ (147) Cancella lo schermo di testo.

COS

– Ritorna il coseno per l'angolo di X radianti

COS(X)

Questa funzione ritorna il valore del coseno di X, dove X è un angolo misurato in radianti.

ESEMPIO:

PRINT COS (π)

–1

DEC

– Ritorna il valore decimale di una stringa numerica esadecimale

DEC (stringa esadecimale)

Questa funzione ritorna il valore decimale di una stringa esadecimale.

ESEMPIO:

PRINT DEC ("D020")
53280

F$="F":PRINT DEC(F$)
15

ERR$

– Ritorna la stringa visualizzando una condizione di errore

ERR$(N)

Questa funzione ritorna una stringa visualizzando una condizione di errore. Vedere anche le variabili di sistema EL e ER e l'Appendice A per una lista dei messaggi di errore BASIC.

ESEMPIO:

```
PRINT ERR$(10)
NEXT WITHOUT FOR
```

EXP

– Ritorna il valore dell'approssimazione di e(2.7182813) elevato alla potenza X

EXP (X)

Questa funzione ritorna un valore di e(2.7182813) elevato a X.

ESEMPIO:

```
PRINT EXP(1)
2.7182813
```

FNxx

– Ritorna il valore dalla funzione definita dall'utente

FNxx(X)

Questa funzione ritorna il valore dalla funzione xx definita dall'utente creata con un'istruzione DEF FNxx.

ESEMPIO:

```
10 DEF FNAA(X)=(X-32)*5/9
20 INPUT X
30 PRINT FNAA(X)
RUN
? 40 (? è il prompt di input)
4.44444445
```

FRE

– Ritorna il numero dei byte disponibili in memoria

FRE (X)

dove X è il numero di banco. X=0 per la memorizzazione dei programmi BASIC e X=1 per il controllo dello spazio disponibile per la variabile BASIC.

ESEMPI:

PRINT FRE (0) 48893	Ritorna il numero di byte disponibile per i programmi BASIC.
PRINT FRE (1) 64256	Ritorna il numero di byte disponibile per la memorizzazione della viariabile BASIC.

HEX$

– Ritorna la stringa numerica esadecimale dal numero decimale

HEX$(X)

Questa funzione ritorna una stringa di quattro caratteri contenente la rappresentazione esadecimale del valore X(0<=<65535). La funzione opposta è DEC.

ESEMPIO:

PRINT HEX$(53280)
D020
NOTA: HEX$(0)="0000"

INSTR

– Ritorna la posizione della stringa 1 nella stringa 2

INSTR (stringa 1, stringa 2[,posizione iniziale])

La funzione INSTR ricerca il primo punto in cui compare la stringa 2 all'interno della stringa 1 e ritorna la posizione nella stringa in cui è stata trovata la corrispondenza. Il parametro opzionale per POSIZIONE INIZIALE stabilisce la posizione nella stringa 1 dove inizia la ricerca. Il valore della POSIZIONE INIZIALE deve essere compreso nella gamma da 1 a 255. Se non viene trovata alcuna corrispondenza o se la POSIZIONE INIZIALE ha un valore maggiore della lunghezza della stringa 1, oppure se la stringa 1 è nulla, INSTR ritorna il valore 0. Se la stringa 2 è nulla, INSTR ritorna 0.

ESEMPIO:

PRINT INSTR("COMMODORE 128,"128")
11

INT

– Ritorna la forma intera di un valore rappresentato in virgola mobile

INT(X)

Questa funzione ritorna il valore intero dell'espressione. Se l'espressione è positiva, la parte frazionaria non viene indicata. Se l'espressione è negativa, viene ritornato il numero più basso successivo.

ESEMPI:

PRINT INT(3.14)
3

PRINT INT(–3.14)
–4

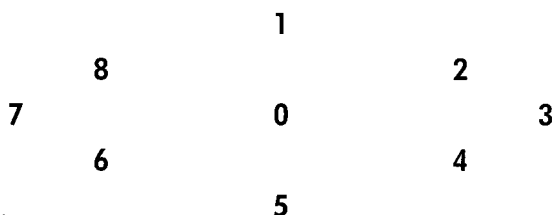
JOY

– Ritorna la posizione del joystick e lo stato del pulsante di fuoco

JOY(N)

dove il valore di N significa:
1 JOY ritorna la posizione del joystick 1.
2 JOY ritorna la posizione del joystick 2.

I valori superiori a 128 compreso indicano che è stato premuto anche il pulsante fuoco. Per trovare il valore di JOY, aggiungere il valore di direzione del joystick più 128 se il pulsante fuoco è premuto. La direzione viene indicata come segue:

		1		
	8		**2**	
7		**0**		**3**
	6		**4**	
		5		

ESEMPI:

JOY (2) is 135

Il joystick 2 spara verso sinistra

IF (JOY (1) AND 128) = 128 THEN PRINT "FUOCO".
Determina se il pulsante fuoco è premuto.

LEFT$

– Ritorna i caratteri all'estrema sinistra nella stringa

LEFT$ (stringa,intero)

Questa funzione ritorna una stringa comprensiva del numero dei caratteri all'estrema sinistra della stringa determinata dal numero intero specificato. L'argomento del numero intero deve essere compreso nella gamma da 0 a 255. Se il numero intero è maggiore della lunghezza della stringa, viene ritornata l'intera stringa. Se viene utilizzato un valore zero, verrà ritornata una stringa nulla (di lunghezza zero).

ESEMPIO:

PRINT LEFT$(COMMODORE",5)
COMMO

LEN

– Ritorna la lunghezza di una stringa

LEN (stringa)

Questa funzione ritorna il numero dei caratteri nell'espressione di stringa. Vengono compresi anche i caratteri non stampabili e gli spazi vuoti.

ESEMPIO:

PRINT LEN("COMMODORE128")
12

LOG

– Ritorna il logaritmo naturale di X

LOG(X)

Questa funzione ritorna il logaritmo naturale di X. Il logaritmo naturale è il logaritmo in base e (vedere EXP(X)).Per convertire in logaritmo in base 10, dividere per LOG(10).

ESEMPIO:

PRINT LOG (37/5)
2.00148

MID$

– Ritorna una sottostringa da una stringa più ampia oppure sovrappone una sottostringa in una stringa più ampia

MID$(stringa,posizione iniziale[,lunghezza])

Questa funzione ritorna una sottostringa specificata dalla LUNGHEZZA, a partire dal carattere specificato dalla posizione iniziale. La posizione iniziale della sottostringa definisce il primo carattere da cui inizia la sottostringa. La lunghezza della sottostringa viene specificata dall'argomento della lunghezza. Entrambi gli argomenti numerici possono avere valori compresi tra 0 e 255. Se il valore della posizione iniziale è maggiore della lunghezza della stringa, oppure se il valore della lunghezza è zero, MID$ ritornerà un valore di stringa nulla. Se l'argomento della lunghezza viene omesso, verranno ritornati tutti i caratteri a destra della posizione iniziale.

ESEMPIO:

PRINT MID$("COMMODORE 128,3,5)
MMODO

ESEMPIO con la funzione di sovrapposizione:

A$="123456":MID$(A$,3,2)="ABCDE":PRINT A$
12 AB56

NOTA: La funzione di sovrapposizione può essere utilizzata per espandere la lunghezza di una stringa, per cui l'esempio precedente MID$(A$,3,5) non è possibile.

PEEK

– Ritorna il contenuto di una locazione di memoria specificata

PEEK(X)

Questa funzione ritorna il contenuto della locazione di memoria X, dove X è situato nella gamma da 0 a 65535, che ritorna un risultato compreso tra 0 e 255. Questo è il contrario dell'istruzione POKE. I dati verranno ritornati dal banco selezionato dall'ultimo comando BANK. Vedere il comando BANK.

ESEMPIO:

```
10 BANK 15:VIC = DEC("D000")
20 FOR I = TO 47
30 PRINT PEEK(VIC + I),
40 NEXT
```

Questo esempio visualizza il contenuto dei registri del chip VIC.

PEN

– Ritorna le coordinate X e Y della penna ottica

PEN(n)

dove n=0 PEN ritorna la coordinata X della posizione della penna ottica.
n=1 PEN ritorna la coordinata Y della posizione della penna ottica.
n=2 PEN ritorna la coordinata X della visualizzazione a 80 colonne.
n=3 PEN ritorna la coordinata Y della visualizzazione a 80 colonne.
n=4 PEN ritorna il valore di innesco della penna ottica.

Notare che, come avviene per le coordinate di sprite, il valore PEN non è scalato ed utilizza coordinate reali e non coordinate grafiche a matrice di punti. La posizione X è data come un numero che va da 60 a 320 circa, mentre la posizione della coordinata Y può essere un numero compreso tra 50 e 250. Queste sono le gamme delle coordinate visibili sullo schermo, mentre tutti gli altri valori non sono visibili. Un valore zero per entrambe le posizioni significa che la penna ottica è fuori schermo e che non ha innescato un interrupt dall'ultima lettura. Notare che per utilizzare PEN non è necessario che COLLISION sia attivato. Per dare maggiore effetto alla penna ottica viene utilizzato uno sfondo bianco. I valori di PEN variano da sistema a sistema.

Al contrario dello schermo a 40 colonne (VIC), le coordinate per le 80 colonne (8563) sono posizioni di carattere riga/colonna e non coordinate di pixel come avviene per lo schermo VIC. Entrambi i valori delle coordinate di schermo per 40 e 80 colonne sono approssimativi e variano a seconda della natura della penna ottica. I valori di lettura non sono validi finchè PEN(4) è zero.

ESEMPI:

10 PRINT PEN(0);PEN(1)	Visualizza le coordinate X e Y della penna ottica.
10 DO UNTIL PEN(4):LOOP	Assicura la validità dei valori di lettura.

```
20 X=PEN(2)
30 Y=PEN(3)
40 REM:PARTE RIMANENTE DEL PROGRAMMA
```

π

– Ritorna il valore di p greco (3.14159265)

π

ESEMPIO:

PRINT π
3.14159265

POINTER

– Ritorna l'indirizzo del nome di variabile

POINTER (nome variabile)

ESEMPIO:

PRINT POINTER(Z)	Questo esempio ritorna l'indirizzo della variabile Z.

POS

– Ritorna la posizione di colonna corrente del cursore all'interno della finestra di schermo dove è posizionato il cursore

POS(X)

La funzione POS indica la posizione del cursore all'interno della finestra di schermo definita. X è un argomento fittizio che deve essere specificato, ma il cui valore viene ignorato.

ESEMPIO:

PRINT "0123456789" POS(1)
012345678910

Questo visualizza la posizione corrente del cursore all'interno della finestra di schermo definita, in questo caso 10.

POT

– Ritorna il valore del potenziometro del paddle di gioco

POT(n)

dove:

n=1, POT ritorna la posizione del paddle #1.
n=2, POT ritorna la posizione del paddle #2.
n=3, POT ritorna la posizione del paddle #3.
n=4, POT ritorna la posizione del paddle #4.

I valori per POT sono compresi nella gamma da 0 a 255. Tutti i valori al di sopra e compreso 256 indicano che è stato premuto anche il pulsante fuoco.

ESEMPIO:

```
10 PRINT POT(1)
20 IF POT(1) > = 256 THEN PRINT "FUOCO"
```

Questo esempio visualizza il valore del paddle di gioco 1.

NOTA: Viene ritornato un valore 255 se non è collegato alcun paddle.

RCLR

– Ritorna il colore della sorgente di colore

RCLR(N)

Questa funzione ritorna il colore (da 1 a 16) assegnato alla sorgente di colore N (0<N<6), dove vengono applicati i seguenti valori di N:

0 = sfondo a 40 colonne
1 = primo piano a matrice di punti
2 = multicolor 1
3 = multicolor 2
4 = cornice a 40 colonne
5 = colore carattere a 40 o 80 colonne
6 = colore di sfondo a 80 colonne

Il contrario della funzione RCLR è il comando COLOR.

ESEMPIO:

```
10 FOR I=0 TO 6
20 PRINT "SORGENTE";I;"È IL CODICE COLORE";RCLR(I)
30 NEXT
```

Questo esempio stampa i codici di colore per tutte le sette sorgenti di colore.

RDOT

– Ritorna la posizione corrente o la sorgente di colore del cursore pixerl

RDOT(N)

dove:

N = 0 ritorna la coordinata X del cursore pixel
N = 1 ritorna la coordinata Y del cursore pixel
N = 2 ritorna la sorgente di colore del cursore pixel

Questa funzione ritorna la locazione della posizione corrente del cursore pixel (CP) oppure la sorgetne di colore corrente del cursore pixel.

ESEMPI:

PRINT RDOT(0)	Ritorna la posizione X del CP
PRINT RDOT(1)	Ritorna la posizione Y del CP
PRINT RDOT(2)	Ritorna la sorgente di colore del CP (da 0 a 3)

RGR

– Ritorna la modalità grafica corrente

RGR(X)

Questa funzione ritorna la modalità grafica corrente. X è un argomento fittizio che deve essere specificato. Il contrario della funzione RGR è il comando GRAPHIC. Il valore ritornato da RGR(X) appartiene alle seguenti modalità:

VALORE	MODALITÀ GRAFICA
0	Testo a 40 colonne (VIC)
1	Matrice di punti standard
2	Matrice di punti a schermo diviso
3	Matrice di punti multicolore
4	Matrice di punti multicolore a schermo diviso
5	Testo a 80 colonne (8563)

ESEMPIO:

PRINT RGR(0)
1

Visualizza la modalità grafica corrente, in questo caso, la modalità standard a matrice di punti.

RIGHT$

– Ritorna la sottostringa dell'estremità destra della stringa

RIGHT$ (<stringa>,<numero>)

Questa funzione ritorna una sottostringa prelevata dai caratteri all'estrema destra dell'argomento di stringa. La lunghezza della sottostringa viene definita dalla lunghezza dell'argomento che può essere un qualsiasi numero intero compreso tra 0 e 255. Se il valore dell'espressione numerica è zero, verrà ritornata una stringa nulla. Se il valore specificato nell'argomento di lunghezza è maggiore della lunghezza della stringa, verrà ritornata l'intera stringa. Vedere anche le funzioni LEFT$ e MID$.

ESEMPI:

PRINT RIGHT$("BASEBALL",5)
EBALL

RND

– Ritorna un numero casuale

RND(X)

Questa funzione ritorna un numero casuale x, come 0<=x<1. Questo può risultare utile nei giochi, per simulare il lancio di un dado ed altri scopi casuali. Viene inoltre utilizzata in alcune applicazioni statistiche.

If X = 0	RND ritorna un numero casuale basato sul clock dell'hardware.
If X > 1	RND genera un numero riproducibile pseudo-casuale basato sul valore di seme.
If X < 0	produce un numero casuale utilizzato come base chiamato seme.

Per simulare un dado che rotola, utilizzare la formula INT(RND(1)*6+1). Per prima cosa il numero casuale viene moltiplicato per 6, ampliando così la gamma a 0-6 (per la precisione minore di sei). Quindi viene aggiunto 1, portando la gamma da 1 a minore di 7. La funzione INT tronca tutte le posizioni decimali, lasciando un risultato compreso nella gamma da 1 a 6.

ESEMPI:

PRINT RND (0) .507824123	Visualizza un numero casuale.
PRINT INT(RND(1)*100+1) 89	Visualizza un numero casuale positivo minore di 100.

RSPCOLOR

– Ritorna i valori degli sprite multicolore

RSPCOLOR(registro)

Dove:

X = 1 RSPCOLOR ritorna lo sprite multicolor 1
X = 2 RSPCOLOR ritorna lo sprite multicolor 2

Il valore di colore ritornato è un valore tra 1 e 16. Il contrario della funzione RSPCOLOR è l'istruzione SPRCOLOR. Vedere anche l'istruzione SPRCOLOR.

ESEMPIO:

```
10 SPRITE 1,1,2,0,1,1,1
20 SPRCOLOR 5,7
30 PRINT"LO SPRITE MULTICOLOR 1 È";RSPCOLOR(1)
40 PRINT"LO SPRITE MULTICOLOR 2 È";RSPCOLOR(2)
RUN

LO SPRITE MULTICOLOR 1 È 5
LO SPRITE MULTICOLOR 2 È 7
```

In questo esempio la riga 10 attiva lo sprite 1, lo colora di bianco, lo espande nelle direzioni X e Y e lo visualizza in modalità multicolore. La riga 20 seleziona gli sprite multicolor 1 e 2. Le righe 30 e 40 stampano i valori RSPCOLOR per multicolor 1 e 2.

RSPPOS

– Ritorna i valori di velocità e posizione di uno sprite

RSPPOS(numero sprite,X)

dove "numero di sprite" identifica quale sprite verrà preso in considerazione e "X" specifica le coordinatee X e Y o la velocità dello sprite.

Quando X è:

0 RSPPOS ritorna la posizione X corrente dello sprite specificato.
1 RSPPOS ritorna la posizione Y corrente dello sprite specificato.
2 RSPPOS ritorna la velocità (da 0 a 15) dello sprite specificato.

ESEMPIO:

```
10 SPRITE 1,1,2
20 MOVSPR 1,45#13
30 PRINT RSPPOS(1,0);RSPPOS(1,1);RSPPOS(1,2)
```

Questo esempio ritorna le coordinate X e Y correnti di sprite e la velocità (13).

RSPRITE

– Ritorna le caratteristiche di sprite

RSPRITE(numero sprite,caratteristiche)

RSPRITE ritorna le caratteristiche di sprite specificate nel comando SPRITE. "Numero di sprite" specifica lo sprite che si sta prendendo in considerazione e "caratteristica" specifica le qualità di visualizzazione dello sprite, come mostrato di seguito:

Caratteristica	RSPRITE ritorna questi valori:		
0	Attivato(1) / Disattivato(0)		
1	Colore dello sprite (da 1 a 16)		
2	Gli sprite vengono visualizzati davanti (0) o dietro (1) agli oggetti sullo schermo		
3	Espansione in direzione X	sì=1,	no=0
4	Espansione in direzione Y	sì=1,	no=0
5	Multicolor	sì=1,	no=0

ESEMPIO:

```
10 FOR I = 0 TO 5
20 PRINT RSPRITE(1,I)
30 NEXT
```

Questo esempio stampa tutte le 6 caratteristiche dello sprite 1.

RWINDOW

– Ritorna le dimensioni della finestra corrente

RWINDOW(n)

Quando n corrisponde a:

0 RWINDOW ritorna il numero di righe della finestra corrente.
1 RWINDOW ritorna il numero di colonne della finestra corrente.
2 RWINDOW ritorna i valori di 40 o 80, a seconda del formato di output corrente utilizzato.

Il contrario della funzione RWINDOW è il comando WINDOW.

ESEMPIO:

```
10 WINDOW 1,1,10,10
20 PRINT RWINDOW(0);RWINDOW(1);RWINDOW(2)
RUN
9940
```

Questo esempio ritorna il numero di righe (9) e di colonne (9) della finestra corrente. Nell'esempio si sottintende che si sta utilizzando il formato a 40 colonne.

SGN

– Ritorna il segno dell'argomento X

SGN(X)

Questa funzione ritorna il segno (positivo, negativo o zero) di X. Il risultato è + 1 se X>0, 0 se X=0 e –1 se X<0.

ESEMPIO:

PRINT SGN(4.5);SGN(0);SGN(–2.3)
10–1

SIN

– Ritorna il seno di un argomento

SIN(X)

Questa è la funzione trigonometrica di seno. Il risultato è il seno di X. X viene misurato in radianti.

ESEMPIO:

PRINT SIN (π/3)
.866025404

SPC

– Salta gli spazi sullo schermo

SPC(X)

Questa funzione viene utilizzata nei comandi PRINT o PRINT# per controllare la formattazione di dati, come output verso lo schermo o output verso un file logico. Il numero di SPC specificati da X determina il numero di caratteri da riempire con spazi sullo schermo o in un file. Per i file su schermo o nastro, il valore dell'argomento è compreso nella gamma da 0 a 255 e per i file disco il valore massimo è 254. Per i file stampante, se viene introdotto uno spazio con SPC nell'ultima posizione di carattere di una riga, si verificherà un ritorno carrello e un avanzamento riga in fase di stampa. Sulla riga successiva non verrà stampato alcuno spazio.

ESEMPIO:

PRINT "COMMODORE";SPC(3);"128"
COMMODORE 128

SQR

– Ritorna la radice quadrata di un argomento

SQR(X)

Questa funzione ritorna il valore della radice quadrata di X, dove X è un numero positivo o 0. Il valore dell'argomento non deve essere negativo, altrimenti verrà visualizzato il messaggio di errore BASIC:?ILLEGAL QUANTITY.

ESEMPIO:

PRINT SQR(25)
5

STR$

– Ritorna la rappresentazione stringa di un numero

STR$(X)

Questa funzione ritorna la rappresentazione stringa di un valore numerico dell'argomento X. Quando il valore STR$ viene convertito, i numeri visualizzati saranno preceduti e seguiti da uno spazio, ad eccezione dei numeri negativi che saranno preceduti da un segno meno. Il contrario della funzione STR$ è la funzione VAL.

ESEMPIO:

PRINT STR$(123.45)
123.45

PRINT STR$(–89.03)
–89.03

PRINT STR$(1E20)
1E+20

TAB

– Sposta il cursore alla posizione di tabulazione nell'istruzione presente

TAB(X)

Questa funzione sposta il cursore in avanti, se possibile, su una posizione relativa dello schermo di testo fornita dall'argomento X, a partire dalla posizione all'estrema sinistra della riga corrente. Il valore dell'argomento può essere compreso tra 0 e 255. Se la posizione di stampa corrente è già oltre la posizione X, il comando TAB verrà ignorato. La funzione TAB può essere utilizzata solo con l'istruzione PRINT in quanto avrà effetti diversi se utilizzata con PRINT# in un file logico a seconda del dispositivo utilizzato.

ESEMPIO:

10 PRINT"COMMODORE"TAB(25)"128"
COMMODORE   128

TAN

– Ritorna la tangente di un argomento

TAN(X)

Questa funzione ritorna la tangente di X, dove X è un angolo misurato in radianti.

ESEMPIO:

PRINT TAN(.785398163)
1

USR

– Chiama la sotto-funzione definita dall'utente

USR(X)

Quando viene utilizzata questa funzione, il programma salta ad un programma in linguaggio macchina il cui punto di inizio è contenuto nelle locazioni di memoria 4633($1219) e 4634($121A), (o 785($0311) e 786($0312) per la modalità C64). Il parametro X viene passato al programma di linguaggio macchina nell'accu-

mulatore di virgola mobile. Viene ritornato un valore al programma BASIC attraverso la variabile chiamata. Per poter ricevere il valore dell'accumulatore di virgola mobile si dovrà ridigere il valore nella variabile del programma. Se la variabile non viene specificata verrà visualizzato un messaggio ILLEGAL QUANTITY ERROR. Questo permette all'utente di scambiare una variabile tra codice oggetto e BASIC.

ESEMPIO (solo per 128):

```
10 POKE 4633,0
20 POKE 4634,192
30 A=USR(X)
40 PRINT A
```

NOTA: Banco di default per 128: 15

Posizionare la locazione iniziale ($C000=49152:$00=0:$C0=192) della routine di linguaggio macchina nella locazione 4633 e 4634. La riga 30 memorizza il valore ritornato dall'accumulatore di virgola mobile.

VAL

– Ritorna il valore numerico di una stringa numerica

VAL(X$)

Questa funzione converte la stringa X$ in un numero (operazione inversa di STR$). La stringa viene controllata dal carattere all'estrema sinistra fino a quello all'estrema destra (tutti i caratteri in formato numerico riconoscibile). Se il Commodore 128 trova caratteri illegali, verrà convertita solo la parte di stringa fino a quel punto. Se non sono presenti caratteri numerici, VAL ritorna a 0.

ESEMPIO:

```
10 A$ = "120"
20 B$ = "365"
30 PRINT VAL (A$) + VAL (B$)
RUN
485
```

XOR

– Ritorna un OR esclusivo

XOR(n1,n2)

Questa funzione fornisce l'OR esclusivo dei valori n1 e n2.

x = XOR (n1,n2)

dove n1 e n2 sono 2 valori senza segno (da 0 a 65535).

ESEMPIO:

PRINT XOR(128,64)
192

NOTA: Non è necessario che n1 e n2 siano numeri interi.

SEZIONE 19

Variabili

Il Commodore 128 utilizza tre tipi di variabili in BASIC: numerica normale, numerica intera e stringa (alfanumerica).

Le VARIABILI NUMERICHE normali, chiamate anche variabili in virgola mobile, possono avere qualsiasi valore esponente da −10 a +10, con un massimo di 9 posizioni decimali. Quando un numero diventa più lungo di nove cifre, il computer lo visualizza in notazione scientifica, con il numero di una cifra e otto posizioni decimali, seguito dalla lettera E e dalla potenza di 10 per la quale viene moltiplicato il numero. Ad esempio, il numero 12345678901 viene visualizzato 1,23456789E + 10.

Le VARIABILI INTERE possono essere utilizzate quando il numero è compreso tra +32767 e −32768, senza posizioni decimali. Una variabile intera è un numero come 5, 10 o −100. I numeri interi occupano meno spazio delle variabili in virgola mobile, specialmente se utilizzati in una matrice.

Le VARIABILI STRINGA sono le variabili utilizzate nei dati di carattere e possono contenere numeri, lettere e qualsiasi altro carattere visualizzabile dal Commodore 128. Un esempio di variabile stringa potrebbe essere "COMMODORE 128".

I NOMI DI VARIABILE possono essere composti da una sola lettera, da una lettera seguita da un numero o da due lettere. I nomi di varibili possono essere più lunghi di due caratteri, ma saranno significativi per il computer solo i primi due. Un numero intero viene specificato utilizzando il simbolo di percento (%) dopo il nome di variabile. Le variabili stringa hanno un simbolo di dollaro ($) che segue il loro nome.

ESEMPI:

Nomi di variabili numeriche: A, A5, BZ
Nomi di variabili intere: A%, A5%, BZ%
Nomi di variabili stringa: A$, A5$, BZ$

Le MATRICI sono liste di variabili con lo stesso nome che utilizzano un numero (o numeri) addizionale per specificare un elemento dalla matrice stessa. Le matrici vengono definite utilizzando l'istruzione DIM e possono essere matrici in virgola mobile, intere o variabili stringa. Il nome variabile di matrice è seguito da una coppia di parentesi () che racchiudono il numero della variabile nella lista.

ESEMPIO:

A(7), BZ%(11),A$(87)

Le matrici possono avere più di una dimensione. Può essere visualizzata una matrice bidimensionale con righe e colonne, utilizzando il primo numero che identifica la riga ed il secondo numero che identifica la colonna (come quando si specifica una particolare griglia su una mappa).

ESEMPIO:

A(7,2),BZ%(2,3,4),Z$(3,2)

I NOMI VARIABILI RISERVATI sono nomi riservati per l'uso da parte del Commodore 128 e non possono essere utilizzati per altri scopi. Questi nomi sono le variabili DS, DS$, ER, ERR$, EL, ST, TI e TI$. Le PAROLE CHIAVE come ad esempio TO e IF o altri nomi che contengono PAROLE CHIAVE, come ad esempio RUN, NEW o LOAD non possono essere utilizzate.

ST è una variabile di stato per input e output (ad eccezione delle normali operazioni di schermo e tastiera). Il valore di ST dipende dal risultato dell'ultima operazione di I/O. Generalmente, se il valore di ST è 0, l'operazione è corretta.

TI e TI$ sono variabili collegate al clock in tempo reale incorporato nel Commodore 128. Questo clock di sistema viene aggiornato ogni sessantesimo di secondo. All'accensione del Commodore 128 il clock parte da 0 e viene ripristinato solo modificando il valore di TI$. La variabile TI fornisce il valore corrente del clock in sessantesimi di secondo. TI$ è una stringa che legge il valore del clock in tempo reale come orologio a 24 ore. I primi due caratteri di TI$ costituiscono l'ora, il terzo ed il quarto carattere i minuti ed il quinto e sesto carattere i secondi. Questa variabile può essere impostata ad un qualsiasi valore (a patto che tutti i caratteri siano numerici) e verrà aggiornata automaticamente come un orologio a 24 ore.

ESEMPIO:

TI $ = "101530"	Imposta il clock sulle ore 10:15 e 30 secondi

Allo spegnimento del Commodore 128 il valore del clock verrà annullato e ripartirà da zero alla prossima accensione del Commodore 128. Il clock tornerà a zero quando il valore supererà 235959 (ore 23, 59 minuti e 59 secondi).

La variabile DS legge il canale di comando del disk drive e ritorna lo stato corrente del drive. Per stampare questa informazione, utilizzare il comando PRINT DS$. Queste varibili di stato vengono utilizzate dopo un'operazione disco, come DLOAD e DSAVE, per localizzare l'errore indicato dalla spia lampeggiante sul disk drive.

ER, EL e ERR$ sono variabili utilizzate nelle routine di identificazione errori e sono generalmente utili solo all'interno del programma. ER ritorna l'ultimo errore incontrato da quando è iniziata l'esecuzione del programma. EL è la riga in cui è stato riscontrato l'errore. ERR$ è una funzione che permette al programma di stampare uno dei messaggi di errore BASIC. PRINT ERR$(ER) stampa il corretto messaggio di errore.

Operatori

Gli OPERATORI BASIC comprendono gli OPERATORI ARITMETICI, RELAZIONALI e LOGICI. Gli operatori ARITMETICI comprendono i simboli seguenti:

+ **addizione**
– **sottrazione**
* **moltiplicazione**
/ **divisione**
↑ **elevazione a potenza (esponenziazione)**

Su una riga in cui compaiono più operatori, le diverse operazioni saranno eseguite in un ordine particolare. Quando vengono utilizzati più operatori nella stessa operazione, il computer assegnerà le priorità nel modo seguente: per prima verrà eseguita l'esponenziazione, quindi moltiplicazione e divisione, e infine addizione e sottrazione. Quando due operatori hanno la stessa priorità, i calcoli verranno effettuati nell'ordine da sinistra a destra. Quando si desidera eseguire le operazioni in un ordine diverso, il BASIC del Commodore 128 offre la possibilità di assegnare ad un calcolo una priorità più alta quando questo viene racchiuso tra parentesi. Le operazioni tra parentesi saranno eseguite per prime. Assicurarsi che nelle equazioni sia stato inserito un numero pari di parentesi aperte e chiuse, altrimenti verrà visualizzato un messaggio SYNTAX ERROR all'esecuzione del programma.

Esistono inoltre operatori per uguaglianze e disuguaglianze, chiamati operatori RELAZIONALI. Gli operatori arimetici hanno sempre la priorità sugli operatori relazionali.

=	**è uguale a**
<	**è minore di**
>	**è maggiore di**
<= **oppure** =<	**è minore di o uguale a**
>= **oppure** =>	**è maggiore di o uguale a**
<> **oppure** ><	**è diverso da**

Infine, esistono tre operatori LOGICI con minore priorità rispetto agli operatori aritmetici e relazionali:

AND
OR
NOT

Questi operatori vengono spesso utilizzati per riunire formule multiple nelle istruzioni IF...THEN. Quando vengono utilizzati con operatori artimetici, vengono calcolati per ultimi (cioè dopo + e –).
Se la relazione indicat nell'espressione è vera, al risultato verrà assegnato il valore intero –1. Se invece è falsa, verrà assegnato il valore 0.

ESEMPI:

IF A=B AND C=D THEN 100	Richiede le condizioni A=B e C=D.
IF A=B OR C=D THEN 100	Richiede una delle due condizioni A=B, C=D o entrambe.
A=5:B=4:PRINT A=B	Visualizza il valore 0.
A=5:B=4:PRINT A>3	Visualizza il valore –1.
PRINT 123 AND 15:PRINT 5 OR 7	Visualizza 11 e 7.

SEZIONE 20

Parole di sistema riservate (Parole chiave)

In questa sezione vengono elencate le parole ed i simboli utilizzati nel linguaggio BASIC 7.0. Queste parole e simboli non possono essere utilizzati all'interno di un programma se non come componenti del linguaggio BASIC. L'unica eccezione è che possono essere utlizzati tra virgolette in un'istruzione PRINT.

ABS
AND
APPEND
ASC
ATN
AUTO
BACKUP
BANK
BEGIN
BEND
BLOAD
BOOT
BOX
BSAVE
BUMP
CATALOG
CHAR
CHR$
CIRCLE
CLOSE
CLR
CMD
COLLECT
COLLISION
COLOR
CONCAT
CONT
COPY
COS
DATA
DCLEAR
DCLOSE
DEC
DEF
DELETE
DIM
DIRECTORY
DLOAD
DO
DOPEN
DRAW
DS
DSAVE
DS$
DVERIFY
EL
ELSE
END
ENVELOPE
ER
ERR$
EXIT
EXP
FAST
FETCH
FILTER
FN
FOR
FRE
GET
GETKEY
GET#
GO64
GOSUB
GOTO
GO TO
GRAPHIC
GSHAPE
HEADER
HELP
HEX$
IF
INPUT
INPUT#
INSTR
INT
JOY
KEY
LEFT$
LEN
LET
LIST
LOAD
LOCATE
LOG
LOOP
MID$
MONITOR
MOVSPR
NEW
NEXT
NOT
OFF*
ON
OPEN
OR
PAINT
PEEK
PEN
PLAY
POINTER
POKE
POS
POT
PRINT
PRINT#
PUDEF
QUIT*
RCLR
RDOT
READ
RECORD
REM
RENAME
RENUMBER
RESTORE
RESUME
RETURN
RGR
RIGHT$
RND
RREG
RSPCOLOR
RSPPOS
RSPRITE
RUN
RWINDOW
SAVE
SCALE
SCNCLR
SCRATCH
SGN
SIN
SLEEP
SLOW
SOUND
SPC(
SPRCOLOR
SPRDEF
SPRITE
SPRSAV
SQR
SSHAPE
ST
STASH
STEP
STOP
STR$
SWAP
SYS
TAB(
TAN
TEMPO
THEN
TI
TI$
TO
TRAP
TRON
TROFF
UNTIL
USING
USR
VAL
VERIFY
VOL
WAIT
WHILE
WIDTH
WINDOW
XOR

* OFF e QUIT sono in fase di sviluppo

Simboli di sistema riservati

I caratteri seguenti sono simboli di sistema riservati.

Simbolo	Utilizzo(i)
+ Simbolo più	Addizione aritmetica; concatenamento di stringa; spostamento dello sprite relativo; dichiarazione di numero decimale nel monitor del linguaggio macchina.
– Simbolo meno	Sottrazione aritmetica; numero negativo; meno unario; spostamento dello sprite relativo.
* Asterisco	Moltiplicazione aritmetica.
/ Barra	Divisione aritmetica.
↑ Freccia verso l'alto	Elevazione a potenza.
Spazio	Separa le parole chiave ed i nomi di variabile.
= Simbolo uguale	Assegnazione del valore; prova di relazione.
< Minore di	Prova di relazione.
> Maggiore di	Prova di relazione.
, Virgola	Formatta l'output nelle liste variabili; parametri funzionali comando/istruzione.
. Punto	Punto decimale nelle costanti a virgola mobile.
; Punto e virgola	Formatta l'output nelle liste variabili.
: Due punti	Separa le istruzioni BASIC multiple su una riga di programma.
" " Virgolette	Racchiude le costanti stringa.
? Punto interrogativo	Abbreviazione della parola chiave PRINT.
(Parentesi aperta	Valutazione e funzioni di espressioni.
) Parentesi chiusa	Valutazione e funzioni di espressioni.
% Percento	Dichiara un nome di variabile come numero intero; dichiara il numero binario nel monitor del linguaggio macchina.
# Numero	Precede il numero di file logico nelle istruzioni di input/output.
$ Simbolo dollaro	Dichiara un nome di variabile come stringa e dichiara un numero esadecimale nel monitor del linguaggio macchina.

&	E commerciale	Dichiara un numero ottale nel monitor del linguaggio macchina.
π	P greco	Dichiara la costante numerica, appross. 3.14159265.

APPENDICI

APPENDICI

APPENDICE A
MESSAGGI DI ERRORE DEL LINGUAGGIO BASIC

I seguenti messaggi di errore vengono visualizzati dal BASIC. I messaggi di errore possono essere visualizzati anche utilizzando la funzione ERR$(). I numeri di errore riportati di seguito si riferiscono solo al numero assegnato all'errore per l'utilizzo con la funzione ERR$().

ERRORE#	NOME DELL'ERRORE	DESCRIZIONE
1	TOO MANY FILES	(TROPPI FILE) Il limite dei file aperti contemporaneamente è 10.
2	FILE OPEN	(FILE APERTO) E' stato fatto un tentativo di aprire un file utilizzando il numero di un file già aperto.
3	FILE NOT OPEN	(FILE NON APERTO) Il numero di file specificato in una istruzione I/O deve essere aperto prima di utilizzarlo.
4	FILE NOT FOUND	(FILE NON TROVATO) Non esiste alcun file con quel nome (disco) o è stato letto un indicatore di fine nastro (nastro).
5	DEVICE NOT PRESENT	(DISPOSITIVO NON PRESENTE) Il dipositivo I/O richiesto non è disponibile o i buffer non sono allocati (cassetta). Verificare che il dispositivo sia collegato ed acceso.
6	NOT INPUT FILE	(FILE NON DI INPUT) E' stato fatto un tentativo per ottenere o introdurre dati da un file che è stato specificato di solo output.
7	NOT OUTPUT FILE	(FILE NON DI OUTPUT) E' stato fatto un tentativo di trasmettere dati ad un file che è stato specificato di solo input.

8	MISSING FILE NAME	(NOME DI FILE MANCANTE) Nel comando manca il nome di file.
9	ILLEGAL DEVICE NUMBER	(NUMERO DI DISPOSITIVO ILLEGALE) E stato fatto un tentativo di utilizzare non correttamente un dispositivo (istruzione SAVE applicata allo schermo, ecc.)
10	NEXT WITHOUT FOR	(NEXT SENZA FOR) I loop non sono nidificati correttamente o c'è un nome di variabile in una istruzione NEXT che non corrisponde a quello contenuto in un'istruzione FOR.
11	SYNTAX	(SINTASSI) Un'istruzione non è riconoscibile dal BASIC. Questo può accadere a causa della mancanza di parentesi, di una parola chiave mal formulata, ecc.
12	RETURN WITHOUT GOSUB	(RETURN SENZA GOSUB) Un'istruzione RETURN è stata incontrata quando nessuna istruzione GOSUB era attiva.
13	OUT OF DATA	(DATI ESAURITI) E' stata incontrata una istruzione READ quando tutti i dati erano stati già letti.
14	ILLEGAL QUANTITY	(QUANTITÀ ILLEGALE) Un numero usato come argomento di una funzione o di una istruzione è al di fuori della gamma permessa.
15	OVERFLOW	Il risultato di un calcolo è maggiore del numero massimo permesso (1.70141833E + 38).

16	OUT OF MEMORY	(MEMORIA ESAURITA) Non c'è più spazio per il codice di programma e/o le variabili di programma, oppure sono state attivate troppe istruzioni nidificate DO, FOR o GOSUB.
17	UNDEF'D STATEMENT	(ISTRUZIONE NON DEFINITA) Il numero di riga voluto non esiste nel programma.
18	BAD SUBSCRIPT	(INDICE NON VALIDO) Il programma ha cercato di indirizzare un elemento di una matrice fuori dalla gamma specificata dall'istruzione DIM.
19	REDIM'D ARRAY	(MATRICE RIDIMENSIONATA) Una matrice può essere dimensionata una sola volta.
20	DIVISION BY ZERO	(DIVISIONE PER ZERO) La divisione per zero non è ammessa.
21	ILLEGAL DIRECT	(MODALITÀ DIRETTA ILLEGALE) Le istruzioni INPUT o GET, oppure INPUT# o GET# sono permesse solo all'interno di un programma.
22	TYPE MISMATCH	(ERRORE DI BATTITURA) Questo si verifica quando un valore numerico viene utilizzato al posto di una stringa o viceversa.
23	STRING TOO LONG	(STRINGA TROPPO LUNGA) Una stringa può contenere un massimo di 255 caratteri.
24	FILE DATA	(DATI DI FILE) Dati non corretti letti da un nastro o da un file disco.

25	FORMULA TOO COMPLEX	(FORMULA TROPPO COMPLESSA) Il computer non era in grado di comprendere questa espressione. Semplificare l'espressione (dividerla in due parti oppure utilizzare un numero minore di parentesi).
26	CAN'T CONTINUE	(NON È POSSIBILE CONTINUARE) Il comando CONT non funziona se il programma non è stato eseguito, se è stato riscontrato un errore o se una riga è stata modificata.
27	UNDEF'D FUNCTION	(FUNZIONE NON DEFINITA) Si è fatto riferimento ad una funzione definita dall'utente che però non è mai stata definita.
28	VERIFY	(VERIFICA) Il programma su nastro o disco non corrisponde al programma in memoria.
29	LOAD	(CARICAMENTO) Si è verificato un problema di caricamento. Riprovare.
30	BREAK	(INTERRUZIONE) È stato premuto il tasto STOP per interrompere l'esecuzione del programma.
31	CAN'T RESUME	(RECUPERO IMPOSSIBILE) Si è incontrata un'istruzione RESUME senza un'istruzione TRAP attiva.
32	LOOP NOT FOUND	(LOOP NON TROVATO) Il programma ha incontrato una istuzione DO e non riesce a trovare il LOOP corrispondente.

33	LOOP WITHOUT DO	(LOOP SENZA DO) Si è incontrato un LOOP senza un'istruzione DO attiva.
34	DIRECT MODE ONLY	(SOLO MODALITÀ DIRETTA) Questo comando è permesso solo in modalità diretta, non da un programma.
35	NO GRAPHICS AREA	(AREA NON GRAFICA) Si è incontrato un comando (DRAW, BOX ecc.) per la creazione di grafici prima che fosse eseguito il comando GRAPHIC.
36	BAD DISK	(DISCO DIFETTOSO) Si è tentato di riformattare un dischetto non formattato o difettoso con il metodo di formattazione veloce (senza ID).
37	BEND NOT FOUND	(BEND NON TROVATO) Il programma ha incontrato "IF...THEN BEGIN" o "IF...THEN...ELSE BEGIN" senza una parola chiave BEND a completamento di BEGIN.
38	LINE # TOO LARGE	(NUMERO DI RGA TROPPO GRANDE) Si è verificato un errore di rinumerazione di programma BASIC. I parametri forniti risultano in un numero di riga > 63999, per cui la rinumerazione non è stata effettuata.
39	UNRESOLVED REFERENCE	(RIFERIMENTO ERRATO) Si è verificato un errore nella rinumerazione di un programma BASIC. Il numero di riga indicato da un comando (es. GOTO 999) non esiste. Quindi la numerazione non è eseguita.

40	UNIMPLEMENTED COMMAND	(COMANDO NON ESEGUITO) È stato incontrato un comando non riconoscibile dal BASIC 7.0.
41	FILE READ	(ERRORE DI LETTURA DEL FILE) È stato riscontrato un errore durante il caricamento o la lettura di un programma o di un file dal disk drive (es. è stato aperto lo sportello del drive mentre era in corso l'operazione di caricamento programma).

APPENDICE B
MESSAGGI DI ERRORE DOS

I seguenti messaggi di errore DOS vengono ritornati attraverso le variabili DS e DS$. La variabile DS contiene solo il numero di errore, mentre le variabili DS$ contiene il numero di errore, il messaggio di errore ed il numero di traccia e di settore corrispondente. NOTA: I numeri dei messaggi di errore inferiori a 20 devono essere ignorati ad eccezione di 01, che fornisce informazioni sul numero dei file cancellati per mezzo del comando SCRATCH.

NUMERO ERRORE	MESSAGGIO DI ERRORE E DESCRIZIONE
20	READ ERROR (ERRORE DI LETTURA) (intestazione del blocco non trovata) Il controller del disco non può localizzare l'intestazione del blocco di dati richiesto. Ciò può essere provocato da un numero di settore illegale, o dal fatto che l'intestazione è stata distrutta.
21	READ ERROR (nessun carattere di sincronizzazione) Il controller del disco non trova un indicatore di sincronizzazione sulla traccia desiderata. Questo può essere provocato da un disallineamento della testina di lettura/scrittura, dalla mancanza del dischetto, da un dischetto non formattato o inserito non correttamente. Può anche indicare un guasto hardware.
22	READ ERROR (blocco dati non presente) Al controller del disco è stato richiesto di leggere o verificare un blocco di dati che non è stato scritto correttamente. Questo messaggio d'errore si può presentare in seguito a comandi BLOCK e può indicare una traccia e/o una richiesta di settore illegale.
23	READ ERROR (errore di cheksum nel blocco di dati) Questo messaggio di errore indica che è stato riscontrato un errore in uno o più byte di dati. I dati sono stati letti nella memoria DOS, ma il checksum sui dati non è corretto. Questo messaggio può anche indicare problemi di messa a terra hardware.

24 READ ERROR
(errore di decodifica byte)
I dati o l'intestazione sono stati letti nella memoria DOS, ma si è verificato un errore hardware causato da una configurazione di bit non valida nel byte di dati. Questo messaggio può anche indicare problemi di messa a terra hardware.

25 WRITE ERROR (ERRORE DI SCRITTURA)
(errore di verifica di scrittura)
Questo messaggio viene visualizzato se il controller rileva un'incongruenza tra i dati scritti ed i dati presenti nella memoria DOS.

26 WRITE PROTECT ON (PROTEZIONE IN SCRITTURA ATTIVATA)
Viene visualizzato questo messaggio quando al controller viene richiesto di scrivere un blocco di dati quando l'interruttore di protezione in scrittura è premuto o se si utilizza un dischetto che è stato protetto in scrittura con l'etichetta relativa.

27 READ ERROR
(errore di cheksum nell'intestazione)
Viene visualizzato questo messaggio quando viene riscontrato un errore di cheksum nell'intestazione del blocco di dati richiesto. Il blocco non è stato letto nella memoria DOS.

28 WRITE ERROR
(blocco dati troppo lungo)
Viene visualizzato questo messaggio quando un blocco di dati è troppo lungo e copre l'indicatore di sincronizzazione dell'intestazione successiva.

29 DISK ID MISMATCH (ERRORE DI IDENTIFICAZIONE DEL DISCO)
Viene visualizzato questo messaggio quando al controller è stato richiesto di accedere ad un dischetto non inizializzato. Il messaggio può essere visualizzato anche se il dischetto possiede una intestazione non corretta.

30	SYNTAX ERROR (ERRORE DI SINTASSI) (sintassi generale) Il DOS non può interpretare il comando inviato al canale di comando. In genere questo è provocato da un numero illegale di nomi di file, o da configurazioni utilizzate in modo errato, ad esempio, se due nomi di file appaiono a sinistra del comando COPY.
31	SYNTAX ERROR (comando non valido) Il DOS non riconosce il comando. Il comando deve essere introdotto per primo nella riga di comando.
32	SYNTAX ERROR (riga troppo lunga) Il comando è più lungo di 58 caratteri. Utilizzare comandi disco abbreviati.
33	SYNTAX ERROR (nome di file non valido) La corrispondenza di configurazione è stata utilizzata in modo errato nel comando OPEN o SAVE. Riscrivere il nome del file in modo corretto.
34	SYNTAX ERROR (nessun file dato) Il nome di file non è stato introdotto nel comando oppure il DOS non lo riconosce come tale. Generalmente si è dimenticato di introdurre nel comando un simbolo due punti (:).
39	SYNTAX ERROR (comando non valido) Questo messaggio viene visualizzato se il canale di comando (indirizzo secondario 15) non viene riconosciuto dal DOS.
50	RECORD NOT PRESENT (RECORD NON PRESENTE) Risultato della lettura del disco al di là dell'ultimo record per mezzo dei comandi INPUT# o GET#. Questo messaggio sarà visualizzato dopo il posizionamento su un record oltre la fine del file in un file relativo. Se però lo scopo era di ampliare il file aggiungendo nuovi record (con un comando PRINT#), ignorare il messaggio di errore. Non si do-

vranno utilizzare i comandi INPUT# e GET# dopo il rilevamento di questo errore senza prima effettuare un riposizionamento.

51	OVERFLOW IN RECORD (OVERFLOW DI RECORD) L'istruzione PRINT# supera il limite del record. L'informazione viene troncata. Dato che il ritorno carrello, che viene trasmesso come indicatore di fine record viene calcolato nella dimensione del record, questo messaggio si presenterà se l'insieme dei caratteri del record (incluso il ritorno carrello finale) supera la dimensione definita.
52	FILE TOO LARGE (FILE TROPPO GRANDE) La posizione del record all'interno di un file relativo indica che si verificherà un overflow di disco.
60	WRITE FILE OPEN (FILE DI SCRITTURA APERTO) Questo messaggio viene visualizzato quando viene aperto per la lettura un file di scrittura che non era stato chiuso.
61	FILE NOT OPEN (FILE NON APERTO) Questo messaggio viene visualizzato quando si tenta di avere accesso ad un file che non era stato aperto nel DOS. A volte in questo caso non viene visualizzato alcun messaggio, viene semplicemente ignorata la richiesta.
62	FILE NOT FOUND (FILE NON TROVATO) Il file richiesto non esiste sul drive specificato.
63	FILE EXISTS (FILE ESISTENTE) Il nome di file del file che si sta creando esiste già sul dischetto.
64	FILE TYPE MISMATCH (ERRORE NELLA SPECIFICA DEL TIPO DI FILE) L'accesso al file richiesto non è possibile se vengono utilizzati file del tipo specificato. Consultare il capitolo concernente il tipo di file.

65 NO BLOCK (NESSUN BLOCCO)
Questo messaggio viene visualizzato insieme all'Allocazione di Blocco. Il settore che si è tentato di allocare è già allocato. I numeri di traccia e di settore visualizzati dal computer sono quelli più alti disponibili. Se il computer indica il numero di traccia (0), tutti i settori rimanenti sono completi. Se il dischetto non è ancora completo, provare una traccia ed un settore più bassi.

66 ILLEGAL TRACK AND SECTOR (SETTORE E TRACCIA ILLEGALI)
Il DOS ha tentato di accedere ad una traccia o ad un blocco che non esistono nel formato utilizzato. Questo messaggio può indicare un problema nella lettura del puntatore sul prossimo blocco.

67 ILLEGAL SYSTEM T OR S (TRACCIA O SETTOPRE DI SISTEMA ILLEGALE)
Questo messaggio di errore speciale indica una traccia o un settore illegale di sistema.

70 NO CHANNEL (nessun canale)
(disponibile)
Il canale richiesto non è disponibile, oppure tutti i canali sono stati utilizzati. Sono disponibili un massimo di cinque buffer per l'utilizzo. Un file sequenziale richiede due buffer; un file relativo ne richiede tre ed il canale errore/comando ne richiede uno. È possibile utilizzare una qualsiasi combinazione di questi purchè non si superino i cinque buffer.

71 DIRECTORY ERROR (ERRORE NELL'ELENCO)
Il BAM (Block Availability Map - mappa di disponibilità blocco) del dischetto non corrispondente alla copia della memoria disco. Per correggere l'errore, inizializzare il dischetto.

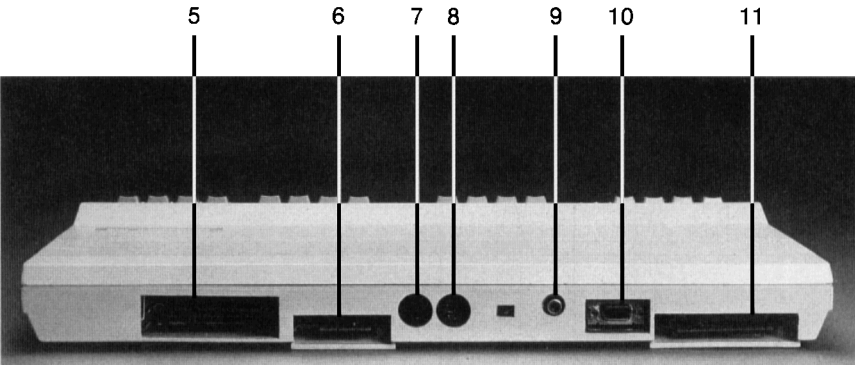
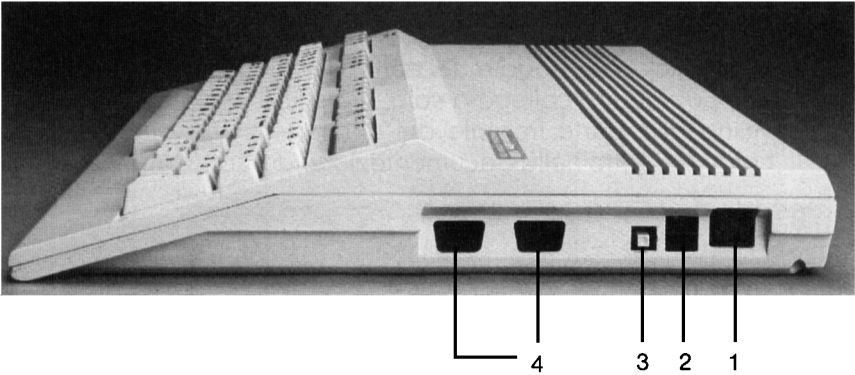
72 DISK FULL (DISCO COMPLETO)
Tutti i blocchi del dishetto sono stati utilizzati oppure lo spazio disponibile nell'elenco è esaurito. Questo messaggio viene visualizzato quando sul dischetto sono disponibili ancora due blocchi per permettere la chiusura del file corrente.

73 DOS MISMATCH (VERSION NUMBER) (INCONGRUENZA DOS - NUMERO VERSIONE)

Il DOS 1 e 2 sono compatibili in lettura ma non in scrittura. I dischi possono essere letti indifferentemente con ambedue i DOS, ma un disco formattato in una determinata versione non può essere scritto nell'altra versione perchè il formato è diverso. Il messaggio di errore viene visualizzato quando si tenta di scrivere su un disco che è stato formattato in un formato non compatibile. Questo messaggio apparirà anche all'accensione, ma in questo caso non indicherà un errore.

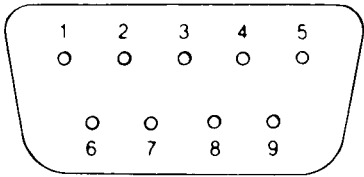
74 DRIVE NOT READY (DRIVE NON PRONTO)

È stato fatto un tentativo di accedere al disk drive senza aver inserito il dischetto; oppure la leva o lo sportello di chiusura del drive è aperto.

APPENDICE C
CONNETTORI/PORTE DEI DISPOSITIVI PERIFERICI

Collegamenti del pannello laterale

1. Presa di corrente - La spina a 5 pin del cavo di alimentazione viene collegata a questa presa.
2. Interruttore di alimentazione - Accendere il computer.
3. Pulsante di Reset - Ripristina il computer (avvio a caldo).
4. Porte del controller - Sono previste due porte del controller numerate 1 e 2, a ciascuna delle quali può essere collegato un joystick o un paddle del controller di gioco. La penna ottica può essere collegata solo alla porta 1, cioè la porta più vicina alla parte frontale del computer. Utilizzare le porte come descritto nella documentazione del software.

Porta di controllo 1

Pin	Tipo	NOTA
1	JOYA0	
2	JOYA1	
3	JOYA2	
4	JOYA3	
5	POT AY	
6	PULSANTE A/LP	
7	+ 5V	Max 50 mA
8	MASSA	
9	POT AX	

Porta di controllo 2

Pin	Tipo	NOTA
1	JOYB0	
2	JOYB1	
3	JOYB2	
4	JOYB3	
5	POT BY	
6	PULSANTE B	
7	+ 5V	Max 50 mA
8	MASSA	
9	POT BX	

Collegamenti posteriori

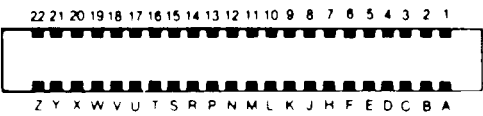
5. Porta di espansione - Questo slot rettangolare è una porta parallela in cui vengono collegate le cartucce di programma o di gioco, oppure interfacce speciali.

Slot di espansione cartuccia

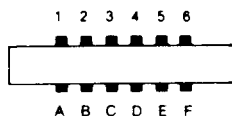
Pin	Tipo
12	BA
13	DMA
14	D7
15	D6
16	D5
17	D4
18	D3
19	D2
20	D1
21	D0
22	MASSA

Pin	Tipo
1	MASSA
2	+5V
3	+5V
4	IRQ
5	R/W
6	CLOCK
7	I/O 1
8	GIOCO
9	EXROM
10	I/O 2
11	ROML

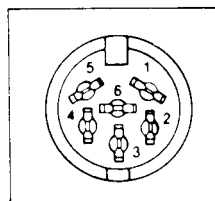
Pin	Tipo
N	A9
P	A8
R	A7
S	A6
T	A5
U	A4
V	A3
W	A2
X	A1
Y	A0
Z	MASSA

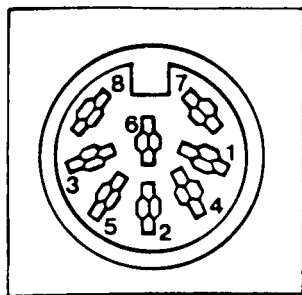
Pin	Tipo
A	MASSA
B	ROMH
C	RESET
D	NMI
E	S 02
F	A15
H	A14
J	A13
K	A12
L	A11
M	A10

6. Porta per la cassetta - In questa porta viene collegato il registratore Datassette 1530 per la memorizzazione di programmi ed informazioni.

Cassetta

Pin	Tipo
A-1	MASSA
B-2	+5V
C-3	MOTORE CASSETTA
D-4	LETTURA CASSETTA
E-5	SCRITTURA CASSETTA
F-6	SENSO CASSETTA

7. Porta seriale - Per mezzo di questa porta è possibile collegare una stampante seriale o un disk drive al Commodore 128.

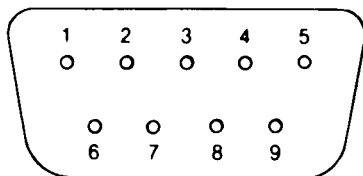
I/O seriale

Pin	Tipo
1	SRQIN
2	MASSA
3	ATN SERIALE IN/OUT
4	CLK SERIALE IN/OUT
5	DATI SERIALI IN/OUT
6	RESET

NOTA: La Porta Seriale Commodore non è compatibile con RS-232. I livelli RS-232 TTL possono essere ottenuti dalla Porta di Utente.

8. Connettore video 40 colonne - Il connettore DIN fornisce audio diretto e segnali video compositi. Questi possono essere collegati al corretto monitor o apparecchiatura audio. Questi segnali possono essere collegati al monitor Commodore oppure utilizzati con componenti separati.

Pin	Tipo	Nota
1	LUM/SYNC	Uscita Luminanza/SYNC
2	MASSA	
3	USCITA AUDIO	
4	USCITA VIDEO	Uscita segnale composito
5	ENTRATA AUDIO	
6	USCITA COLOR	Uscita segnale croma
7	NC	Nessun collegamento
8	NC	Nessun collegamento

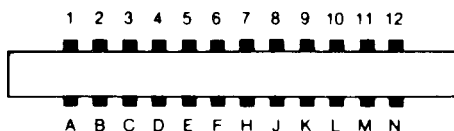
9 Connettore RF - Questo connettore permette di ottenere immagine e audio sull'apparecchio televisivo (Un televisore può visualizzare solo 40 colonne di schermo).

10. Connettore RGBI 80 colonne - Questo connettore a 9 pin emette audio diretto e un segnale RGBI (intensità/rosso/verde/blu).

11. Porta di utente - A questa porta possono essere collegati diversi dispositivi di interfaccia.

Pin	Segnale
1	Massa
2	Massa
3	Rosso
4	Verde
5	Blu
6	Intensità
7	Monocromatico
8	Sincr. orizzontale
9	Sincr. verticale

I/O utente

Pin	Tipo	Nota
1	MASSA	
2	+5V	MAX 100 mA
3	RESET	
4	CNT1	
5	SP1	
6	CNT2	
7	SP2	
8	PC2	
9	ATN SR. IN	
10	9 VAC	MAX 100 mA
11	9 VAC	MAX 100 mA
12	MASSA	

Pin	Tipo	Nota
A	MASSA	
B	FLAG2	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
G	PB4	
H	PB5	
J	PB6	
K	PB7	
L	PA2	
M	MASSA	

APPENDICE D
CODICI DI VISUALIZZAZIONE SCHERMO - 40 colonne

Nella figura riportata di seguito sono elencati tutti i caratteri che fanno parte del set di caratteri per lo schermo Commodore. La figura mostra i numeri da introdurre (POKE) nella memoria di schermo (locazioni da 1024 a 2023) per ottenere un particolare carattere sullo schermo a 40 colonne. Per impostare la memoria colore, utilizzare le locazioni dalla numero 55296 alla numero 56295. Viene inoltre mostrata la corrispondenza dei caratteri ai numeri prelevati dallo schermo (PEEK).

Sono disponibili due set di caratteri. In modalità a 80 colonne sono disponibili simultaneamente i due set, mentre in modalità a 40 colonne se ne può utilizzare uno solo per volta. I set vengono selezionati premendo il tasto SHIFT contemporaneamente al tasto C= (Commodore).

Dal BASIC, l'istruzione PRINT CHR$(142) attiverà la modalità maiuscolo/grafica, mentre PRINT CHR$(14) attiverà la modalità maiuscolo/minuscolo.

Tutti i numeri indicati nella lista possono essere visualizzati in inversione video. Il codice di carattere per l'inversione si ottiene aggiungendo 128 ai valori indicati.

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
@		0	N	n	14	£		28
A	a	1	O	o	15	]		29
B	b	2	P	p	16	↑		30
C	c	3	Q	q	17	←		31
D	d	4	R	r	18	SPACE		32
E	e	5	S	s	19	!		33
F	f	6	T	t	20	"		34
G	g	7	U	u	21	#		35
H	h	8	V	v	22	$		36

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
I	i	9	W	w	23	%		37
J	j	10	X	x	24	&		38
K	k	11	Y	y	25	'		39
L	l	12	Z	z	26	(		40
M	m	13	[		27	)		41
*		42		G	71			100
+		43		H	72			101
,		44		I	73			102
–		45		J	74			103
.		46		K	75			104
/		47		L	76			105
0		48		M	77			106
1		49		N	78			107
2		50		O	79			108
3		51		P	80			109
4		52		Q	81			110
5		53		R	82			111
6		54		S	83			112
7		55		T	84			113
8		56		U	85			114
9		57		V	86			115
:		58		W	87			116
;		59		X	88			117
<		60		Y	89			118
=		61		Z	90			119
>		62			91			120
?		63			92			121
		64			93			122
	A	65			94			123
	B	66			95			124

	C	67	SPACE	96		125
	D	68		97		126
	E	69		98		127
	F	70		99		

I codici da 128 a 255 sono i codici da 0 a 127 in inversione video.

APPENDICE E
CODICI ASCII E CHR$

Questa appendice mostra i caratteri che verranno visualizzati utilizzando il comando PRINT CHR$(X), per tutti i valori possibili di X. Inoltre nell'appendice verranno indicati i valori ottenuti battendo PRINT ASC("x"), dove x è un qualsiasi carattere visualizzabile. Questo è utile per la valutazione del carattere ricevuto in un'istruzione GET, per convertire i caratteri maiuscoli in minuscoli e per stampare comandi basati sui caratteri (come ad esempio l'attivazione della modalità maiuscolo/minuscolo) che non possono essere racchiusi tra virgolette.

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
	0		23	.	46	E	69
	1		24	/	47	F	70
	2		25	0	48	G	71
	3		26	1	49	H	72
	4		27	2	50	I	73
WHT	5	RED	28	3	51	J	74
	6	CRSR →	29	4	52	K	75
	7	GRN	30	5	53	L	76
DISABLES SHIFT C=	8	BLU	31	6	54	M	77
ENABLES SHIFT C=	9	SPACE	32	7	55	N	78
	10	!	33	8	56	O	79
	11	"	34	9	57	P	80
	12	#	35	:	58	Q	81
RETURN	13	$	36	;	59	R	82
SWITCH TO LOWER CASE	14	%	37	<	60	S	83
	15	&	38	=	61	T	84
	16	'	39	>	62	U	85
CRSR ↓	17	(	40	?	63	V	86
RVS ON	18	)	41	@	64	W	87

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
CLR HOME	19	*	42	A	65	X	88
INST DEL	20	+	43	B	66	Y	89
	21	,	44	C	67	Z	90
	22	–	45	D	68	[	91
£	92	♥	115	f4	138		161
]	93		116	f6	139		162
↑	94		117	f8	140		163
←	95		118	SHIFT RETURN	141		164
	96		119	SWITCH TO UPPER CASE	142		165
♠	97	♣	120		143		166
	98		121	BLK	144		167
	99	♦	122	CRSR	145		168
	100		123	RVS OFF	146		169
	101		124	CLR HOME	147		170
	102		125	INST DEL	148		171
	103		126	Brown	149		172
	104		127	Lt. Red	150		173
	105		128	Dk. Gray	151		174
	106	Orange	129	Gray	152		175
	107		130	Lt. Green	153		176
	108		131	Lt. Blue	154		177
	109		132	Lt. Gray	155		178
	110	f1	133		156		179
	111	f3	134		157		180
	112	f5	135		158		181
	113	f7	136		159		182
	114	f2	137	SPACE	160		183

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
	184		186		188		190
	185		187		189		191

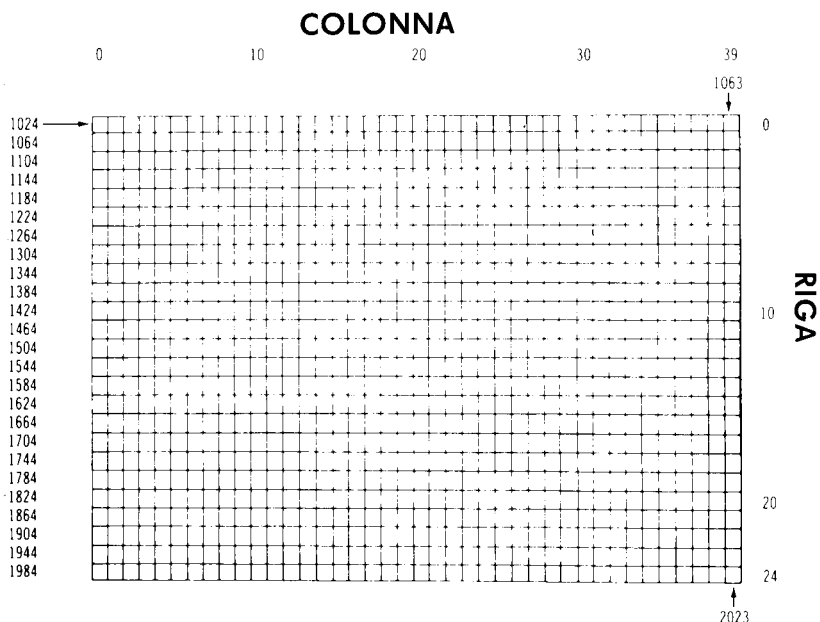
CODICI	**192-233**	**COME**	**96-127**
CODICI	**224-254**	**COME**	**160-190**
CODICE	**255**	**COME**	**126**

NOTA: I codici riportati qui sopra sono per la modalità C64. Per i codici speciali in modalità C128 consultare l'Appendice I.

APPENDICE F
MAPPE DI MEMORIA DI SCHERMO E COLORE - Modalità C128 a 40 colonne e modalità C64

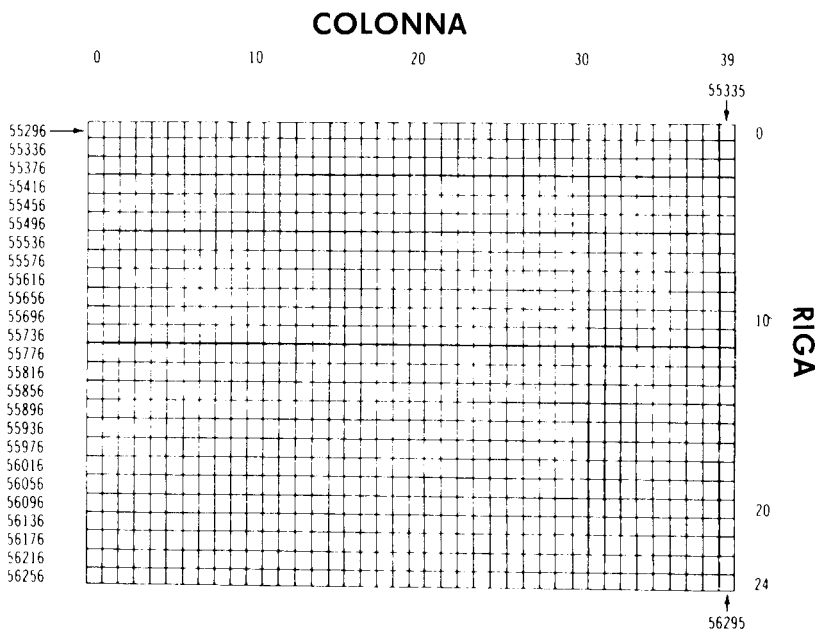
Le mappe riportate di seguito indicano le locazioni di memoria utilizzate in modalità a 40 colonne (C128 e C64) per l'identificazione sullo schermo dei caratteri e del loro colore. Ogni mappa è comandata individualmente e consiste di 1.000 posizioni.

Il carattere visualizzato sulle mappe può essere controllato direttamente per mezzo del comando POKE.

MAPPA DI MEMORIA DELLO SCHERMO

La mappa dello schermo viene ottenuta per mezzo del comando POKE seguito da un valore di codice di visualizzazione schermo (vedere Appendice D). Ad esempio:

POKE 1024,13

visualizzerà la lettera M nell'angolo superiore sinistro dello schermo.

MAPPA DI MEMORIA DEL COLORE

Se la mappa di colore viene visualizzata per mezzo di POKE seguito dal valore di colore, questo causerà una modifica del colore del carattere. Ad esempio:

POKE 55296,1

cambierà il colore della lettera M da verde chiaro a bianco.

Codici di colore - 40 colonne

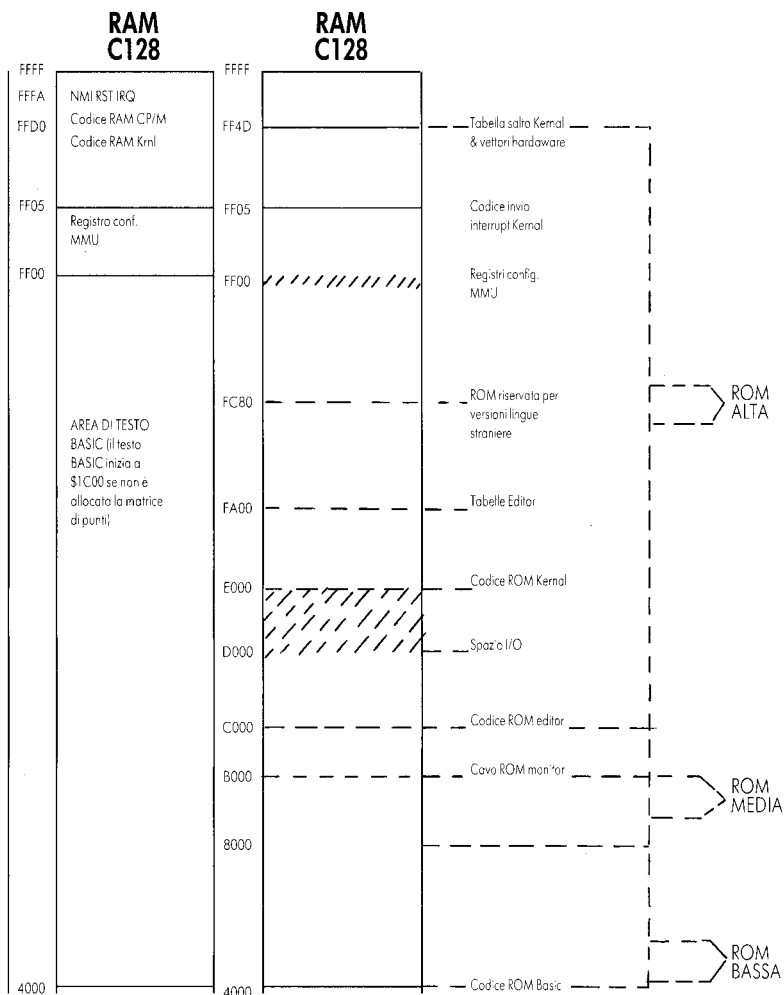
0	Nero	8	Arancio
1	Bianco	9	Marrone
2	Rosso	10	Rosso chiaro
3	Azzurro	11	Grigio scuro
4	Porpora	12	Grigio
5	Verde	13	Verde chiaro
6	Blu	14	Blu chiaro
7	Giallo	15	Grigio chiaro

Memoria di controllo cornice 53280
Memoria di controllo sfondo 53281

APPENDICE G
FUNZIONI TRIGONOMETRICHE DERIVATE

FUNZIONE	EQUIVALENTE IN BASIC
SECANTE	SEC(X)=1/COS(X)
COSECANTE	CSC(X)=1/SIN(X)
COTANGENTE	COT(x)=1/TAN(X)
SENO INVERTITO	ARCSIN(X)=ATN(X/SQR(-X*X+1))
COSENO INVERTITO	ARCCOS(X)=-ATN(X/SQR (-X*X+1) =+/2
SECANTE INVERTITA	ARSEC(X)=ATN(X/SQR(X*X-1))
COSECANTE INVERTITA	ARCCSC(X)=ATN(X/SQR(X*X-1)) +(SGN(X)-1*π/2
COTANGENTE INVERTITA	ARCOT(X)=ATN(X)+π/2
SENO IPERBOLICO	SINH(X)=EXP(X)-EXP(-X))/2
COSENO IPERBOLICO	COSH(X)=(EXP(X)+EXP(-X))/2
TANGENTE IPERBOLICA	TANH(X)=EXP(-X)/(EXP(x)+EXP(-X))*2+1
SECANTE IPERBOLICA	SECH(X)=2/(EXP(X)=EXP(-X))
COSECANTE IPERBOLICA	CSCH(X)=2/(EXP(X)-EXP(-X))
COTANGENTE IPERBOLICA	COTH(X)=EXP(-X)/(EXP(X)-EXP(-X))*2+1
SENO IPERBOLICO INVERTITO	ARCSINH(X)=LOG(X+SQR(X*X+1))
COSENO IPERBOLICO INVERTITO	ARCCOSH(X)=LOG(X+SQR(X*X-1))
TANGENTE IPERBOLICA INVERTITA	ARCTANH(X)=LOG((1+X)/(1-X))/2
SECANTE IPERBOLICA INVERTITA	ARCSECH(X)=LOG((SQR (-X*X+1)+1/X)
COSECANTE IPERBOLICA INVERTITA	ARCCSCH(X)=LOG((SGN(X)*SQR (X*X+1)/X)
COTANGENTE IPERBOLICA INVERTITA	ARCCOTH(X)=LOG((X+1)/(X-1))/2

APPENDICE H
MAPPA DI MEMORIA DI SISTEMA

MAPPA DI MEMORIA DEL COMMODORE 128

MAPPA DI MEMORIA MODALITÀ C128

Indirizzo	RAM C128	ROM C128
4000		4000
4000–2000	Schermo VIC a matrice di punti	
2000–1C00	Schermo VIC a colori (Vm#2)	
1C00, 1B00, 1A00, 1900	Riservato per software tasti funzione	
1800–1400	Riservato per sistemi con lingue straniere	
1400–1300		
1300–1200	Variabili Basic assolute	
1200–1108	Variabili Basic DOS/VSP	
1108–1100	Codice Reset CP/M	
1100–1000	Buffer tasto funzione	
1000, 0F00, 0E00	Area definizione sprite	
0E00–0D00	Buffer output RS-232	
0D00–0C00	Buffer intput RS-232	
0C00–0BC0	(pag. caric. disco)	
0BC0–0B00	Buffer cassetta	
0B00–0A00	Variabili assolute monitor e Kernal	
0A00, 0900, 0800	Tempo di esecuzione stack Basic	
0800–0400	Schermo di testo VIC (VM#1)	
0400–0380	Codice RAM Basic	
0380–033C	Tabelle Kernal	
033C–02FC	Indiretti	
02FC–02A2	Codice RAM Kernal	
02A2–0200	Buffer input Basic e monitor	
0200–0149	Stack di sistema	
0149–0110	Utilizzo Basic DOS	
0110–0100	BUFFER F	
0100–0090	Kernal Z.P.	
0090–0002	Basic Z.P.	
0002–0000		

APPENDICE I
CODICE CONTROL E ESC

Codici di controllo

CHR$	Sequenza tasto	Funzione	Modalità: C64	C128
CHR$(2)	CTRL B	Sottolineatura (80)		*
CHR$(5)	CTRL 2 o CTRL E	Imposta il carattere sul colore bianco (40) e (80)	*	*
CHR$(7)	CTRL G	Emette segnale acustico		*
CHR$(8)	CTRL H	Disattiva la modifica del set di caratteri	*	
CHR$(9)	CTRL I	Attiva la modifica del set di caratteri	*	
		Sposta il cursore al punto di tabulazione successivo		*
CHR$(10)	CTRL J	Invia un ritorno carrello con avanzamento riga	*	
		Invia un avanzamento riga		*
CHR$(11)	CTRL K	Disattiva la modifica del set di caratteri		*
CHR$(12)	CTRL L	Attiva la modifica del set di caratteri		*
CHR$(13)	CTRL M	Invia al computer un ritorno carrello ed un avanzamento riga ed introduce una riga di BASIC	*	*
CHR$(14)	CTRL N	Imposta il set di caratteri su maiuscolo/minuscolo	*	*
CHR$(15)	CTRL O	Attiva il lampeggiamento (80)		*
CHR$(17)	CRSR VERSO IL BASSO o CTRL Q	Sposta il cursore di una riga verso il basso	*	*
CHR$(18)	CTRL 9	Stampa i caratteri invertiti	*	*
CHR$(19)	HOME o CTRL S	Sposta il cursore alla posizione home (angolo superiore sinistro) dello schermo (finestra corrente)	*	*

NOTA: (40)... solo visualizzazione a 40 colonne
(80)... solo visualizzazione a 80 colonne

CHR$	Sequenza tasto	Funzione	Modalità: C64	C128
CHR$(20)	DEL/CTRL T	Cancella l'ultimo carattere battuto e sposta di uno spazio verso sinistra tutti i caratteri a destra del carattere cancellato.	*	*
CHR$(24)	CTRL X	Imposta/annulla i punti di tabulazione		
CHR$(27)	ESC o CTRL [	Invia un carattere ESC		*
CHR$(28)	CTRL 3 o CTRL /	Imposta il colore di carattere su rosso (40) e (80)	*	*
CHR$(29)	CRSR o CTRL]	Sposta il cursore di una colonna verso destra	*	*
CHR$(30)	CTRL 6 o CTRL ↑	Imposta il colore di carattere su verde (40) e (80)	*	*
CHR$(34)	"	Stampa un simbolo di virgolette sullo schermo ed imposta l'editor in modalità quote	*	*
CHR$(129)	C= 1	Imposta il colore di carattere su arancio (40); porpora scuro (80)	*	*
CHR$(130)		Disattiva la sottolineatura (80)		*
CHR$(131)		Esegue un programma. Questo codice CHR$ non funziona con PRINT CHR$(131) ma dal buffer della tastiera	*	*
CHR$(133)	F1	Codice CHR$ riservato per il tasto F1	*	*
CHR$(134)	F3	Codice CHR$ riservato per il tasto F3	*	*
CHR$(135)	F5	Codice CHR$ riservato per il tasto F5	*	*

NOTA: (40)... Solo visualizzazione a 40 colonne
(80)... solo visualizzazione a 80 colonne

CHR$	Sequenza tasto	Funzione	Modalità: C64	C128
CHR$(136)	F7	Codice CHR$ riservato per il tasto F7	*	*
CHR$(137)	F2	Codice CHR$ riservato per il tasto F2	*	*
CHR$(138)	F4	Codice CHR$ riservato per il tasto F4	*	*
CHR$(139)	F6	Codice CHR$ riservato per il tasto F6	*	*
CHR$(140)	F8	Codice CHR$ riservato per il tasto F8	*	*
CHR$(141)	SHIFT RETURN	Invia un ritorno carrello ed un avanzamento riga senza introdurre una riga BASIC	*	*
CHR$(142)		Imposta il set di caratteri su maiuscolo/grafico	*	*
CHR$(143)		Disattiva il lampeggiamento (80)		*
CHR$(144)	CTRL 1	Imposta il colore del carattere su nero (40) e (80)	*	*
CHR$(145)	CRSR SU	Sposta il cursore o la posizione di stampa in alto di una riga	*	*
CHR$(146)	CTRL 0	Termina la visualizzazione di campo invertito	*	*
CHR$(147)	HOME	Cancella la finestra e sposta il cursore nell'angolo superiore sinistro dello schermo	*	*

NOTA: (40)... Solo visualizzazione a 40 colonne
(80)... solo visualizzazione a 80 colonne

CHR$	Sequenza tasto	Funzione	Modalità: C64	C128
CHR$(148)	INST	Imposta il colore del carattere su grigio scuro (40); azzurro scuro (80)	*	*
CHR$(149)	C= 2	Imposta il colore del carattere su grigio (40) e (80)	*	*
CHR$(150)	C=	Imposta il colore del carattere su verde chiaro (40) e (80)	*	*
CHR$(151)	C= 4	Imposta il colore del carattere su blu chiaro (40) e (80)	*	*
CHR$(152)	C= 5	Imposta il colore del carattere su grigio chiaro (40) e (80)	*	*
CHR$(153)	C= 6	Imposta il colore del carattere su porpora (40) e (80)	*	*
CHR$(154)	C= 7	Sposta il cursore a sinistra di una colonna	*	*
CHR$(155)	C= 8	Imposta il colore del carattere su azzurro (40); azzurro chiaro (80)	*	*
CHR$(156)	CTRL 5	Sposta i caratteri a partire dalla posizione del cursore a destra di una colonna	*	*
CHR$(157)	CRSR SINISTRA	Imposta il colore del carattere su marrone (40); giallo scuro (80)	*	*
CHR$(158)	CTRL 4	Imposta il colore del carattere su rosso chiaro (40) e (80)	*	*

NOTA: (40)... Solo visualizzazione a 40 colonne
(80)... solo visualizzazione a 80 colonne

Codici ESC

Di seguito vengono fornite le sequenze delle funzioni ESC disponibili sul Commodore 128. Le sequenze ESC vengono introdotte premendo e rilasciando il tasto "ESC" e quindi premendo uno dei tasti elencati di seguito.

FUNZIONE ESC	TASTO ESC
Annulla la modalità quote e inserimento	ESC O
Cancella fino alla fine della riga corrente	ESC Q
Cancella fino all'inizio della riga corrente	ESC P
Cancella fino alla fine dello schermo	ESC @
Sposta all'inizio della riga corrente	ESC J
Sposta alla fine della riga corrente	ESC K
Attiva la modalità auto-inserimento	ESC A
Disattiva la modalità auto-inserimento	ESC C
Cancella la riga corrente	ESC D
Introduce una riga	ESC I
Imposta i punti di tabulazione di default (8 spazi)	ESC Y
Cancella tutti i punti di tabulazione	ESC Z
Attiva lo scorrimento dello schermo	ESC L
Disattiva lo scorrimento dello schermo	ESC M
Scorrimento verso l'alto	ESC V
Scorrimento verso il basso	ESC W
Attiva il segnale acustico (con Control-G)	ESC G
Disattiva il segnale acustico	ESC H
Imposta il cursore sulla modalità di non lampeggiamento	ESC E
Imposta il cursore sulla modalità di lampeggiamento	ESC F
Imposta la parte inferiore della finestra alla posizione del cursore	ESC B
Imposta la parte superiore della finestra alla posizione del cursore	ESC T
Passa dalla visualizzazione a 40 colonne se ci si trova in quella a 80 colonne e alla visualizzazione a 80 colonne se ci si trova in quella a 40 colonne	ESC X

Le sequenze ESC mostrate di seguito sono valide solo per la visualizzazione a 80 colonne. (Vedere la Sezione 8 per dettagli sull'uso della visualizzazione a 80 colonne).

Imposta il cursore nella forma a trattino	ESC U
Imposta il cursore nella forma a quadrato	ESC S
Imposta l'inversione video	ESC R
Riporta lo schermo alla condizione normale (non-inversione)	ESC N

NOTA: (40)... solo visualizzazione a 40 colonne
(80)... solo visualizzazione a 80 colonne

APPENDICE J
MONITOR DEL LINGUAGGIO MACCHINA

INTRODUZIONE

Il Commodore 128 possiede un programma per il monitor incorporato del linguaggio macchina che permette all'utente di scrivere e comprendere facilmente i programmi di linguaggio macchina. Il MONITOR del Commodore 128 è composto da un monitor del linguaggio macchina, da un mini-assemblatore e da un disassemblatore. Il monitor incorporato funziona solo in modalità C128, con 40 o 80 colonne.

I programmi in linguaggio macchina scritti utilizzando il MONITOR del Commodore 128 possono essere usati così come sono, oppure possono essere usati come subroutine molto veloci per i programmi in BASIC. Infatti il MONITOR del Commodore 128 può funzionare con il BASIC.

È necessario posizionare i programmi in linguaggio Assembler nella memoria in modo tale che il programma BASIC non si sovrapponga ad essi.

Per accedere al monitor dal BASIC, battere:

MONITOR RETURN

Comandi del monitor Commodore 128

ASSEMBLE	Assembla una riga del codice 8502
COMPARE	Confronta due sezioni di memoria ed indica le differenze tra una e l'altra
DISASSEMBLE	Disassembla una riga del codice 8502
FILL	Riempie una gamma di memoria con il byte che è stato specificato
GO	Inizia l'esecuzione partendo dall'indirizzo specificato
HUNT	Ricerca nella memoria all'interno di una gamma specificata un set di byte per tutte le volte che questo appare
JUMP	Salta al sottoprogramma
LOAD	Carica un file da nastro o disco
MEMORY	Visualizza i valori esadesimali delle locazioni di memoria

REGISTERS	Visualizza i registri 8502
SAVE	Salva su nastro o disco
TRANSFER	Trasferisce il codice da una sezione di memoria all'altra
VERIFY	Confronta la memoria da nastro o disco
EXIT	Disattiva il MONITOR del Commodore 128
(punto)	Assembla una riga del codice 8502
(maggiore di)	Modifica la memoria
(punto e virgola)	Modifica le visualizzazioni del registro 8502
@	Visualizza lo stato del disco, invia il comando di disco, visualizza l'elenco

Il Commodore 128 visualizza indirizzi esadecimali di 5 cifre all'interno del monitor del linguaggio macchina. Generalmente un numero esadecimale è composto da solo quattro cifre, che rappresentano la gamma di indirizzo permessa. L'ultima cifra supplementare a sinistra specifica la configurazione di BANCO (nel momento in cui il comando indicato viene eseguito). Vedere la tabella di configurazione memoria riportata di seguito:

0-solo RAM 0
1-solo RAM 1
2-solo RAM 2
3-solo RAM 3
4-ROM INT, RAM 0, I/O
5-ROM INT, RAM 1, I/O
6-ROM INT, RAM 2, I/O
7-ROM INT, RAM 3, I/O
8-ROM EST, RAM 0, I/O
9-ROM EST, RAM 1, I/O
A-ROM EST, RAM 2, I/O
B-ROM EST, RAM 3, I/O
C-KERNAL + INT (bassa), RAM 0, I/O
D-KERNAL + EST (bassa), RAM 1, I/O
E-KERNAL + BASIC, RAM 0, ROMCAR
F-KERNAL + BASIC, RAM 0, I/O

Riassunto dei descrittori di campo del monitor

I designatori seguenti precedono i campi di dati del monitor (es. dump di memoria). Se questi designatori vengono utilizzati come comandi, si otterrà la modifica del contenuto della memoria o del registro da parte del monitor per mezzo dei dati specificati.

. <punto> precede le righe di codice disassemblato.
> <parentesi angolare destra> precede le righe del dump di memoria.
; <punto e virogola> precede la riga di un dump di registro.

I designatori seguenti precedono i campi numerici (es. indirizzo) e specificano la radice (base numerica) del valore. Se utilizzati come comandi, questi designatori istruiscono il monitor a visualizzare semplicemente il valore dato in ciascuna delle quattro radici.

<nullo> (default) precede i valori esadecimali.
$ <dollaro> precede i valori esadecimali (base 16).
+ <più> precede i valori decimali (base 10)
& <e commerciale> precede i valori ottali (base 8).
ù <percento> precede i valori binari (base 2).

I caratteri seguenti vengono utilizzati dal monitor come delimitatori di campo o terminatori di riga (tranne che all'interno di una stringa ASCII).

<spazio>, delimitatori - separa due campi.
, <virgola>, delimitatore - separa due campi.
: <due punti>, terminatore - fine logica della riga.
? <punto interrogativo>, terminatore - fine logica della riga.

NOTA: Il simbolo <> racchiude i parametri richiesti.
il simbolo [] racchiude i parametri opzionali.

Descrizione dei comandi del monitor del Commodore 128

Notare che qualsiasi campo numerico (es. indirizzi, numeri di dispositivi e byte di dati) può essere specificato sotto forma di potenze. Questo vale anche per il campo dell'operando del comando ASSEMBLE. Notare inoltre l'aggiunta della sintassi dell'elenco al comando di disco.

Quando si lavora con il monitor, è possibile utilizzare la funzione automatica di messaggi di errore Kernal. Questo significa che il Kernal visualizzerà 'I/O ERROR#' ed il codice di errore ogni volta che si verificherà un errore di I/O dal MONITOR. Questa funzione viene disattivata all'uscita dal MONITOR.

COMANDO: **A**
SCOPO: Introdurre una riga di codice Assembler.
SINTASSI: **A** <indirizzo> <mnemonico codiceop> <operando>

<indirizzo> Un numero indicante la locazione di memoria in cui introdure il codiceop.

<mnemonico codiceop> Un mnemonico di linguaggio Assembler della tecnologia standard MOS, es. LDA, STX, ROR.

<operando> L'operando, quando richiesto, può essere una qualsiasi modalità di indirizzamento permessa.

Viene utilizzato un RETURN per indicare la fine della riga Assembler. Se nella riga sono presenti degli errori, verrà visualizzato un punto interrogativo per indicare l'errore ed il cursore si sposterà sulla riga successiva. Per correggere l'errore o gli errori su quella riga può essere utilizzato lo screen editor.

ESEMPIO:
```
.A01200 LDX #$00
.A01202
```

NOTA: Un punto (.) equivale ad un comando ASSEMBLE.

ESEMPIO:
.02000 LDA #$23

COMANDO: **C**
SCOPO: Confrontare due aree di memoria.
SINTASSI: **C** <indirizzo 1> <indirizzo 2> <indirizzo 3>

<indirizzo 1>	Un numero indicante l'indirizzo iniziale dell'area di memoria da confrontare.
<indirizzo 2>	Un numero esadecimale indicante l'indirizzo finale dell'area di memoria da confrontare.
<indirizzo 3>	Un numero indicante l'indirizzo iniziale dell'altra area di memoria da confrontare. Sullo schermo vengono visualizzati gli indirizzi incongruenti.

COMANDO: **D**
SCOPO: Disassemblare il codice macchina in mnemonici ed operandi di linguaggio Assembler.
SINTASSI: **D** [<indirizzo 1>][<indirizzo 2>]

<indirizzo 1>	Un numero che imposta l'indirizzo per iniziare il disassemblaggio.
<indirizzo 2>	Un indirizzo finale opzionale del codice da disassemblare.

Il formato di disassemblaggio ottenuto in questo modo differisce leggermente dal formato Assembler. La differenza è che il primo carattere di un disassemblaggio è un punto invece che una A e che viene listato anche l'esadecimale del codice.

Un listato di disassemblaggio può essere modificato utilizzando lo screen editor. Eseguire le modifiche dei mnemonici o degli operandi sullo schermo, quindi premere il tasto di ritorno carrello. Con questa operazione la riga viene introdotta e l'Assembler viene richiamato per eseguire ulteriori modifiche.

Un disassemblaggio può essere impaginato. Se si batte D <RETURN>, verrà visualizzata la pagina successiva del disassemblaggio.

ESEMPIO:

```
D 3000 3003
   .03000 A900       LDA #$00
   .03002 FF          ???
   .03003 D0 2B       BNE $3030
```

COMANDO: **F**

SCOPO: Riempire una gamma di locazioni con un byte specificato.

SINTASSI: **F** <indirizzo 1> <indirizzo 2> <byte>

<indirizzo 1>	Prima locazione da riempire con il <byte>.
<indirizzo 2>	Ultima locazione da riempire con il <byte>.
<valore di byte>	Numero da introdurre.

Questo comando viene utilizzato per l'inizializzazione delle strutture di dati o di altre aree RAM.

ESEMPIO:

F 0400 0518 EA

Riempirà le locazioni di memoria da $0400 a $0518 con $EA (istruzione NOP).

COMANDO: **G**
SCOPO: Iniziare l'esecuzione di un programma ad un indirizzo specificato.
SINTASSI: **G** [<indirizzo>]

<indirizzo> Un indirizzo da cui deve iniziare l'esecuzione. Se l'indirizzo viene omesso, l'esecuzione inizierà al PC corrente (Il PC corrente può essere visualizzato per mezzo del comando R).

Il comando GO ripristina tutti i registri (visualizzabili utilizzando il comando R) ed inizia l'esecuzione partendo dall'indirizzo iniziale specificato. Si consiglia di utilizzare il comando GO con cautela. Per ritornare alla modalità MONITOR del Commodore 128 dopo l'esecuzione di un programma in linguaggio macchina, utilizzare l'istruzione BRK alla fine del programma.

ESEMPIO:

G 140C

L'esecuzione inizierà alla locazione $140C.

COMANDO: **H**
SCOPO: Cercare nella memoria all'interno di una gamma specificata un set di byte tutte le volte che questo compare.
SINTASSI: **H** <indirizzo 1> <indirizzo 2> <dati>

<indirizzo 1> Indirizzo iniziale della procedura di ricerca.
<indirizzo 2> Indirizzo finale della procedura di ricerca.
<dati> I dati impostati per la ricerca possono essere numeri o una stringa ASCII.

ESEMPIO:

H A000 A101 A9 FF 4C

Ricerca dei dati $A9, $FF, $4C, da A000 a A101.

H 2000 9800 'CASSA'

Ricerca della stringa alfabetica "CASSA".

COMANDO: **L**
SCOPO: Caricare un file da cassetta o disco.
SINTASSI: **L** <"nomefile">[,<dispositivo>[,indirizzo di arresto caricamento]]

<"nomefile">	Un qualsiasi nome di file permesso per il Commodore 128.
<dispositivo>	Un numero indicante il dispositivo da cui caricare. 1 = cassetta, 8 = disco (oppure 9, A, ecc.).
[indirizzo di arresto caricamento]	Opzione per il caricamento di un file fino a un indirizzo specificato.

Il comando LOAD carica un file in memoria. Se non viene utilizzato l'indirizzo di caricamento alternato, il file verrà caricato all'indirizzo nel banco 0 specificato nell'intestazione della cassetta o nei primi due byte di un file disco. L'indirizzo di caricamento alternato viene utilizzato per specificare un diverso indirizzo iniziale compreso nella gamma da $00000 a $FFFFF.

ESEMPIO:

L "PROGRAMMA",$12000	Carica il file denominato PROGRAMMA dal disco sul banco 1 partendo da $2000.

COMANDO: **M**
SCOPO: Visualizzare la memoria come dump esadecimale e ASCII all'interno della gamma di indirizzo specificata.
SINTASSI: **M** [<indirizzo 1>][<indirizzo 2>]

<indirizzo 1>	Primo indirizzo del dump di memoria. Opzionale. Se questo indirizzo non viene specificato, verrà visualizzata una pagina. Il primo byte è il numero di banco da visualizzare, i quattro byte successivi sono il primo indirizzo da visualizzare.

<indirizzo 2>	Ultimo indirizzo del dump di memoria. Opzionale. Se questo indirizzo non viene specificato, verrà visualizzata una pagina. Il primo byte è il numero di banco da visualizzare, i quattro byte successivi sono l'indirizzo finale da visualizzare.

La memoria viene visualizzata nel formato seguente:

>03000 45 58 2E 56 41 4C 55 45:EX. VALUE

Il contenuto della memoria può essere modificato per mezzo dello screen editor. Spostare il cursore sui dati da modificare, battere la correzione desiderata e premere <RETURN>. In caso di errata locazione RAM o di tentativo di modificare la ROM, viene visualizzato un flag di errore (?). A destra dei dati esadecimali viene visualizzato un dump ASCII dei dati in video invertito (per riconoscerlo dai dati visualizzati sullo schermo). Quando un carattere non può essere stampato, al suo posto verrà visualizzato un punto in inversione video. Come per il comando DISASSEMBLE, lo spostamento dello schermo verso il basso viene effettuato battendo M e <RETURN>.

```
ESEMPIO:
M F41F1 F4201
  >F41F1 20 43 4F 4D 4D 4F 44 4F:COMMODO
  >F41F9 52 45 20 45 4C 45 43 54:RE ELECT
  >F4201 52 4F 4E 49 43 53 2C 20:RONICS,
```

Nota: Quanto sopra viene prodotto dall'editor a 40 colonne.

COMANDO: **R**

SCOPO: Indicare i registri 8502 importanti. Vengono visualizzati: il registro di stato del programma, il contatore di programma, l'accumulatore, i registri di indice X e Y ed il puntatore di stack.

SINTASSI: **R**

ESEMPIO:

```
R
  PC SR AC XR YR SP
; 01002 01 02 03 04 F6
```

NOTA: Può essere utilizzato un simbolo; (punto e virgola) per la modifica delle visualizzazioni di registro, esattamente come > può essere utilizzato per la modifica dei registri di memoria.

COMANDO: **S**

SCOPO: Salvare un'area di memoria su nastro o disco.

SINTASSI: **S** <"nomefile">,<dispositivo>,<indirizzo 1>,<indirizzo 2>

<"nomefile"> Un qualsiasi nome di file permesso per il Commodore 128. Per salvare i dati, il nome di file deve essere racchiuso tra virgolette. Non possono essere utilizzati apici.

<dispositivo> Un numero indicante il dispositivo su cui deve essere portato il file. Cassetta = 1, disco = 8, 9, ecc.

<indirizzo 1> Indirizzo iniziale di memoria da salvare.

<indirizzo 2> Indirizzo finale di memoria da salvare + 1. Tutti i dati fino a ed escluso il byte di dati di questo indirizzo vengono salvati.

Il file può essere richiamato per mezzo del comando L quando si effettua il salvataggio su cassetta sarà possibile salvare solo dal banco 0.

ESEMPIO:

S"GIOCO"8,0400,0C00

Salverà su disco la memoria da $0400 a $0C00.

COMANDO: **T**

SCOPO: Trasferire segmenti di memoria da un'area di memoria all'altra.

SINTASSI: **T** <indirizzo 1> <indirizzo 2> <indirizzo 3>

<indirizzo 1>	Indirizzo iniziale dei dati da spostare.
<indirizzo 2>	Indirizzo finale dei dati da spostare.
<indirizzo 3>	Indirizzo iniziale della nuova locazione su cui verranno spostati i dati.

I dati possono essere spostati dalla memoria alta alla memoria bassa e viceversa. Ulteriori segmenti di memoria di qualsiasi lunghezza possono essere spostati da una memoria all'altra. Viene eseguito un "confronto" automatico durante il trasferimento dei byte e tutte le differenze vengono elencate per indirizzo.

ESEMPIO:

T 1400 1600 1401

Porta i dati da $1400 fino a e compreso $1600 più in alto di un byte nella memoria.

COMANDO: **V**

SCOPO: Confrontare un file su cassetta o disco con il contenuto della memoria.

SINTASSI: **V** <"nomefile">[,<dispositivo>][,indirizzo iniziale]

<"nomefile">	Un qualsiasi nome di file permesso per il Commodore 128.
<dispositivo>	Un numero indicante il dispositivo su cui si trova il file: cassetta = 1, disco = 8, 9, ecc.
[indirizzo iniziale]	Opzione per l'inizio della verifica da questo indirizzo.

Il comando di verifica confronta un file con il contenuto della memoria. Il Commodore 128 visualizza VERIFYING. Se viene riscontrato un errore, apparirà anche la parola ERROR, se invece l'operazione di verifica è positiva, riapparirà il cursore.

ESEMPIO:

V"CARICAMENTO",8

COMANDO: **X**
SCOPO: Ritornare al BASIC.
SINTASSI: **X**

COMANDO: > (maggiore di)
SCOPO: Può essere utilizzato per impostare fino a otto o sedici locazioni.
SINTASSI: > <indirizzo>[<byte dati...8/16>]

<indirizzo>	Primo indirizzo di memoria da impostare.
<byte dati 1...8/16>	Dati da introdurre nelle locazioni di memoria successive dopo l'indirizzo con uno spazio prima di ogni byte dati.

Il numero massimo di byte che si possono introdurre è 8 (nella modalità a 40 colonne) o 16 (nella modalità a 80 colonne). Dopo la pressione del tasto RETURN viene visualizzato il contenuto delle 8/16 locazioni dopo l'indirizzo.

ESEMPI:

>2000	Visualizza la riga di byte dopo $2000
>2000 31 32 38	Introduce i valori a $2000 e visualizza la riga di byte dopo $2000.

COMANDO: @
SCOPO: Può essere utilizzato per visualizzare lo stato del disco.
SINTASSI: @ [<unità #>],<stringa comando disco>

<unità #>	Numero dell'unità (opzionale).
<stringa comando disco>	Stringa di comando disco.

NOTA: Utilizzando solo il simbolo @ si ottiene lo stato del disk drive.

ESEMPI:

@ 00,OK,00,00	controlla lo stato del disco
@,I	Inizializza il drive 8
@,$	Visualizza l'elenco del drive 8.

APPENDICE K
ABBREVIAZIONI BASIC 7.0

NOTA: Le abbreviazioni riportate di seguito funzionano in modalità maiuscolo/grafica. Premere il tasto o i tasti alfabetici indicati, quindi tenere premuto il tasto SHIFT contemporaneamente alla lettera indicata dopo SHIFT.

PAROLA CHIAVE	ABBREVIAZIONE
ABS	A SHIFT B
APPEND	A SHIFT P
ASC	A SHIFT S
ATN	A SHIFT T
AUTO	A SHIFT U
BACKUP	BA SHIFT C
BANK	B SHIFT A
BEGIN	B SHIFT E
BEND	BE SHIFT N
BLOAD	B SHIFT L
BOOT	B SHIFT O
BOX	nessuna
BSAVE	B SHIFT S
BUMP	B SHIFT U
CATALOG	C SHIFT A
CHAR	CH SHIFT A
CHR$	C SHIFT H
CIRCLE	C SHIFT I
CLOSE	CL SHIFT O
CLR	C SHIFT L
CMD	C SHIFT M
COLLECT	COLL SHIFT E
COLINT	nessuna
COLLISION	COL SHIFT L
COLOR	COL SHIFT O
CONCAT	C SHIFT O
CONT	nessuna
COPY	CO SHIFT P
COS	nessuna
DATA	D SHIFT A
DEC	nessuna
DCLEAR	DCL SHIFT E

PAROLA CHIAVE	ABBREVIAZIONE
DCLOSE	D SHIFT C
DEF FN	nessuna
DELETE	DE SHIFT L
DIM	D SHIFT I
DIRECTORY	DI SHIFT R
DLOAD	D SHIFT L
DO	nessuna
DOPEN	D SHIFT O
DRAW	D SHIFT R
DSAVE	D SHIFT S
DVERIFY	D SHIFT V
EL	nessuna
END	nessuna
ENVELOPE	E SHIFT N
ER	nessuna
ERR$	E SHIFT R
EXIT	EX SHIFT I
EXP	E SHIFT X
FAST	nessuna
FETCH	F SHIFT E
FILTER	F SHIFT I
FOR	F SHIFT O
FRE	F SHIFT R
FNxx	nessuna
GET	G SHIFT E
GETKEY	GETK SHIFT E
GET#	nessuna
GOSUB	GO SHIFT S
GO64	nessuna
GOTO	G SHIFT O
GRAPHIC	G SHIFT R
GSHAPE	G SHIFT S
HEADER	HE SHIFT A
HELP	HE SHIFT L
HEX$	H SHIFT E
IF...GOTO	nessuna
IF...THEN...ELSE	nessuna
INPUT	nessuna
INPUT#	I SHIFT N
INSTR	IN SHIFT S
INT	nessuna
JOY	J SHIFT O

PAROLA CHIAVE	ABBREVIAZIONE
KEY	K SHIFT E
LEFT$	LE SHIFT F
LEN	nessuna
LET	L SHIFT E
LIST	L SHIFT I
LOAD	L SHIFT O
LOCATE	LO SHIFT C
LOG	nessuna
LOOP	LO SHIFT O
MID$	M SHIFT I
MONITOR	MO SHIFT N
MOVESHAPE	nessuna
MOVSPR	M SHIFT O
NEW	nessuna
NEXT	N SHIFT E
ON...GOSUB	ON...GO SHIFT S
ON...GOTO	ON...G SHIFT O
OPEN	O SHIFT P
PAINT	P SHIFT A
PEEK	PE SHIFT E
PEN	P SHIFT E
PI	nessuna
PLAY	P SHIFT L
POKE	PO SHIFT O
POS	nessuna
POT	P SHIFT O
PRINT	?
PRINT#	P SHIFT R
PRINT USING	?US SHIFT I
PUDEF	P SHIFT U
RBUMP	RB SHIFT U
RCLR	R SHIFT C
RDOT	R SHIFT D
READ	RE SHIFT A
RECORD	R SHIFT E
REM	nessuna
RENAME	RE SHIFT N
RENUMBER	REN SHIFT U
RESTORE	RE SHIFT S
RESUME	RES SHIFT U
RETURN	RE SHIFT T

PAROLA CHIAVE	ABBREVIAZIONE
RGR	R SHIFT G
RIGHT$	R SHIFT I
RLUM	nessuna
RND	R SHIFT N
RREG	R SHIFT R
RSPCOLOR	RSP SHIFT C
RSPPOS	R SHIFT S
RSPR	nessuna
RSPRITE	RSP SHIFT R
RUN	R SHIFT U
RWINDOW	R SHIFT W
SAVE	S SHIFT A
SCALE	SC SHIFT A
SCNCLR	S SHIFT C
SCRATCH	SC SHIFT R
SGN	S SHIFT G
SIN	S SHIFT Y
SLEEP	S SHIFT L
SLOW	nessuna
SOUND	S SHIFT O
SPC(	nessuna
SPRCOLOR	SPR SHIFT C
SPRDEF	SPR SHIFT D
SPRITE	S SHIFT P
SPRSAV	SPR SHIFT S
SQR	S SHIFT Q
SSHAPE	S SHIFT S
STASH	S SHIFT T
STATUS	nessuna
STEP	ST SHIFT E
STOP	ST SHIFT O
STR$	ST SHIFT R
SWAP	S SHIFT W
SYS	nessuna
TAB(	T SHIFT A
TAN	nessuna
TEMPO	T SHIFT E
TI	nessuna
TI$	nessuna
TO	nessuna

PAROLA CHIAVE	ABBREVIAZIONE
TRAP	T SHIFT R
TROFF	TRO SHIFT F
TRON	TR SHIFT O
UNTIL	U SHIFT N
USR	U SHIFT S
VAL	nessuna
VERIFY	V SHIFT E
VOL	V SHIFT O
WAIT	W SHIFT A
WHILE	W SHIFT H
WIDTH	WI SHIFT D
WINDOW	W SHIFT I
XOR	X SHIFT O

APPENDICE L
RIASSUNTO DEI COMANDI DISCO

Questa appendice elenca i comandi utilizzati per le operazioni di disco nelle modalità C128 e C64 del Commodore 128. Per informazioni più dettagliate su questi comandi, consultatre l'Enciclopedia BASIC 7.0, Capitolo V. Altre informazioni sono riportate sul manuale del disk drive.

I **nuovi** comandi **BASIC 7.0** possono essere utilizzati **solo** in modalità C128. **Tutti** i comandi BASIC 2.0 possono invece essere utilizzati in **entrambe** le modalità.

Comando	**Utilizzo**	**Basic 2.0**	**Basic 7.0**
APPEND	Accodare dati a un file*		*
BLOAD	Caricare un file binario partendo dalla locazione di memoria specificata		*
BOOT	Caricare ed eseguire un programma		*
BSAVE	Salvare un file binario dalla locazione di memoria specificata		*
CATALOG	Visualizzare sullo schermo il contenuto dell'elenco del disco*		*
CLOSE	Chiudere un file logico di disco	*	*
CMD	Reinviare l'output di schermo al file disco	*	*
COLLECT	Liberare lo spazio inaccessibile su disco*		*
CONCAT	Concatenare due file dati*		*
COPY	Copiare i file da un drive all'altro*		*

* Sebbene in BASIC 2.0 non vi sia un singolo comando equivalente a questa funzione, esiste un'istruzione multi-comando equivalente. Vedere il manuale del disk drive a proposito di queste convenzioni del BASIC 2.0.

Comando	Utilizzo	Basic 2.0	Basic 7.0
DCLEAR	Reimpostare e reinizializzare disk drive*		*
DCLOSE	Chiudere i file logici di disco		*
DIRECTORY	Visualizzare sullo schermo l'elenco del contenuto del disco*		*
DLOAD	Caricare un programma BASIC da disco*		*
DOPEN	Aprire un file disco per operazioni di lettura e/o scrittura*		*
DSAVE	Salvare un programma BASIC su disco*		*
DVERIFY	Confrontare un programma in memoria con un programma su disco*		*
GET#	Ricevere l'input da un file disco aperto	*	*
HEADER	Formattare un disco*		*
LOAD	Caricare un file dal disco	*	*
OPEN	Aprire un file per input o output	*	*
PRINT#	Inviare dati a un file	*	*
RECORD	Posizionare i puntatori di file relativi*		*
RENAME	Modificare il nome di un file su disco*		*
RUN nomefile	Eseguire il programma BASIC da disco		*
SAVE	Memorizzare sul disco il programma in memoria	*	*
VERIFY	Confrontare il programma in memoria con il programma su disco.	*	*

* Sebbene in BASIC 2.0 non vi sia un singolo comando equivalente a questa funzione, esiste un'istruzione multi-comando equivalente. Vedere il manuale del disk drive a proposito di queste convensione del BASIC 2.0.

GLOSSARIO

GLOSSARIO

Questo glossario fornisce una breve spiegazione di alcune delle parole più utilizzate nel linguaggio informatico.

Accoppiatore acustico o Modem acustico: Dispositivo che converte i segnali digitali in toni udibili per la trasmissione via telefono. La velocità è limitata a 1.200 baud, o bit per secondo (bps). Vedere Modem a collegamento diretto.

Alfanumerico: Lettere numeri e simboli speciali della tastiera, esclusi i caratteri grafici.

ALU: Aritmetic Logic Unit (Unità Logica Aritmetica). Parte di un'unità centrale di elaborazione (CPU) in cui vengono gestiti i dati binari.

Animazione: Uso delle istruzioni del computer per simulare il movimento di un oggetto sullo schermo tramite spostamenti graduali e progressivi.

ASCII: Acronimo di American Standard Code for Information Interchange (Codice Standard Americano per Scambi di Informazione). Per rappresentare i caratteri alfanumerici viene utilizzato un codice a sette bit. Questo codice è molto utile ad esempio per l'invio di informazioni dalla tastiera al computer, e da un computer all'altro. Vedere Codice di Stringa di Carattere.

Assegnazione, istruzione di: Istruzione BASIC che imposta una variabile, una còstante o un elemento di matrice ad un valore numerico o di stringa specifico.

Assemblatore: Programma che traduce le istruzioni da linguaggio Assembler in istruzioni in linguaggio macchina.

Assembler, linguaggio: Linguaggio orientato alla macchina con il quale i mnemonici vengono utilizzati per rappresentare ogni istruzione in linguaggio macchina. Ogni CPU possiede un proprio linguaggio assembler specifico. Vedere CPU e linguaggio macchina.

Attacco: Velocità con cui una nota musicale raggiunge il volume di picco partendo da zero.

Attivare: Abilitare un bit, un byte o una funzione specifica del computer.

BASIC: Acronimo per Beginner's All-purpose Symbolic Instruction Code.

Baud: Velocità di trasmissione dei dati seriali. In origine era un termine telegrafico, 300 baud equivalgono circa ad una velocità di trasmissione di 30 byte o caratteri per secondo.

Binario: Sistema di numerazione in base 2. Tutti i numeri vengono rappresentati come una sequenza di 0 e 1.

Bit: Abbreviazine di Binary digIT (cifra binaria). Un bit è l'unità più piccola in un computer. Ogni cifra binaria può avere uno o due valori, zero o uno. Un bit è attivato o "on" se è uguale a uno, mentre è disattivato o "off" se è uguale a zero.

Bit, controllo dei: Metodo di trasmissione di dati seriali con cui ciascun bit ha uno speciale significato e ciascun carattere è preceduto e seguito da bit di start e di stop.

Bit di parità: Un 1 o uno 0 aggiunto ad un gruppo di bit che indica se la somma dei bit dà un risultato pari o dispari.

Bit di start: Bit o gruppo di bit che identifica l'inizio di una parola di dati.

Bit di stop: Bit o gruppo di bit che identificano la fine della parola di dati e definiscono lo spazio tra le parole di dati.

Bus: Linee seriali o parallele utilizzate per trasferire i segnali da un dispositivo all'altro. I computer vengono spesso identificati dal tipo di struttura del bus (ad es. computer con bus S-100 ecc.).

Bus, rete: Sistema in cui tutte le stazioni o i dispositivi del computer comunicano per mezzo di un canale di distribuzione comune o bus.

Byte: Gruppo di otto bit che compongono l'unità più piccola di memorizzazione indirizzabile di un computer. Ogni locazione di memoria del Commodore 128 contiene un byte di informazioni. Un byte è l'unità di memorizzazione richiesta per rappresentare un carattere nella memoria. Vedere Bit.

Carattere: Qualsiasi simbolo sulla tastiera del computer che può essere visualizzato sullo schermo. I caratteri comprendono numeri, lettere, simboli grafici e di punteggiatura.

Carattere, memoria di: Area nella memoria del Commodore 128 che memorizza le configurazioni dei caratteri codificate che vengono visualizzate sullo schermo.

Caratteri grafici: Caratteri non alfanumerici sulla tastiera del computer.

Caratteri, set di: Gruppo di caratteri correlati. I set di caratteri del Commodore 128 comprende: lettere maiuscole, lettere minuscole e caratteri grafici.

Cavo piatto: Gruppo di cavi attaccati parallelamente uno all'altro.

Chip: Circuito elettronico miniaturizzato per l'esecuzione delle operazioni con il computer, come ad esempio input/output, produzione di grafici e di suono.

Clock: Circuito di temporizzazione per microprocessore.

Codice di stringa di carattere: Valore numerico assegnato per rappresentare un carattere del Commodore 128 nella memoria del computer.

Codice sorgente: Programma non eseguibile scritto in linguaggio evoluto. Un compilatore o un assemblatore deve tradurre il codice sorgente in codice oggetto (linguaggio macchina) in modo che il computer lo possa comprendere.

Colore della cornice: Colore dei bordi dello schermo.

Colore dello sfondo: Colore della parte di schermo su cui sono posizionati i caratteri.

Comando: Istruzione BASIC utilizzata in modalità diretta per l'esecuzione di un'azione. Vedere Modalità diretta.

Compilatore: Programma che traduce un linguaggio ad alto livello, come il BASIC, in linguaggio macchina.

Computer: Dispositivo elettronico digitale che memorizza ed elabora informazioni.

Condizione(i): Espressione(i) tra le parole IF e THEN, valutata come vera o falsa in un'istruzione IF...THEN. L'istruzione condizinale IF...THEN permette al computer di decidere le operazioni da svolgere.

Contatore: Variabile che indica quante volte si è verificato un evento in un programma.

Controller di interfaccia video (VIC): Il chip MOS 6566 che rende possibile i grafici a 40 colonne con il Commodore 128.

Coordinata: Punto preciso di una griglia con valori verticali (Y) e orizzontali (X).

CPU: Acronimo per Central Processing Unit (Unità Centrale di Elaborazione). Parte del computer contenente i circuiti che controllano ed eseguono le istruzioni del computer.

Crunch: Riduzione della memoria del computer utilizzata per la memorizzazione di un programma.

Cursore: Quadratino lampeggiante che indica la posizione corrente in cui ci si trova sullo schermo.

Data base: Grande quantità di dati memorizzati in modo organizzato. Un sistema di gestione data-base è un programma che permette di accedere alle informazioni.

Datassette: Dispositivo utilizzato per la memorizzazione sequenziale su nastro di programmi e file di dati.

Dati: Numeri, lettere o simboli introdotti nel computer per l'elaborazione.

Dati, pacchetto: Mezzo di trasmissione dati seriali in un pacchetto che comprende una sequenza per l'individuazione di errori.

Decadimento: Velocità alla quale il volume di una nota musicale passa dal valore di picco ad un volume medio definito livello di sostegno. Vedere Sostegno.

Decremento: Diminuire di un valore specifico, una variabile indice o un contatore.

Digitale: Termine relativo alla tecnologia dei computer e della trasmissione dati, dove tutte le informazioni sono codificate come bit attivati/ disattivati (1 oppure 0).

Dimensione: Caratteristica di una matrice che specifica la direzione lungo un asse in cui sono memorizzati gli elementi della matrice. Ad esempio, una matrice bidimensionale ha un asse X per le colonne e un asse Y per le righe. Vedere Matrice.

Disattivare: Disabilitare un bit, un byte o una funzione specifica del computer.

Disk drive: Dispositivo di memorizzazione di massa ad accesso casuale, che memorizza e carica i file da/a un floppy disk.

Dispositivo di collegamento dati: Parte logica del controllo dei dati che assicura che la comunicazione tra i dispositivi adiacenti è priva di errori.

Dispositivo di interfaccia suono (SID): Chip MOS 6581 per la sintesi del suono che gestisce le caratteristiche sonore del Commodore 128.

DOS: Vedere Sistema Operativo su Disco.

Durata: Lunghezza di una nota musicale.

EPROM: PROM che può essere cancellata dall'utente, generalmente esponendola alla luce ultravioletta. Vedere PROM.

Esadecimale: Si riferisce al sistema numerico in base 16. I programmi in linguaggio macchina vengono spesso scritti in notazione esadecimale.

Eseguire: Mettere in esecuzione istruzioni specifiche in un comando o un'istruzione di programma.

Espressione: Combinazione di costanti, variabili o elementi di matrice su cui agiscono operatori logici, matematici o relativi per ottenere un valore numerico.

File: Programma o raccolta di dati memorizzati su disco o nastro come unità.

Firmware: Istruzioni del computer memorizzate nella ROM, come avviene per le cartucce dei giochi.

Forma d'onda: Rappresentazione grafica della forma di un'onda sonora. La forma d'onda determina alcune delle caratteristiche fisiche del suono.

Frequenza: Numero di onde sonore di un tono per secondo. La frequenza corrisponde al passo del tono udibile.

Frequenza portante: Segnale costante trasmesso tra due dispositivi di comunicazione, modulato per codificare le informazioni binarie.

Funzione: Operazione predefinita che dà come risultato un valore singolo.

Generatore di inviluppo: Componente del Commodore 128 che produce inviluppi specifici (attacco, decadimento, sostegno, rilascio) per le note musicali. Vedere Forma d'onda.

Grafica: Immagini visive sullo schermo che rappresentano i dati in memoria (cioè caratteri, simboli e immagini).

Griglia: Matrice bidimensionale divisa in righe e colonne. Le griglie vengono spesso utilizzate per la progettazione di sprite e caratteri programmabili.

Hardware: Componenti fisici di un sistema, come ad esempio tastiera, disk drive e stampante.

Home: Angolo superiore sinistro dello schermo.

IC: Integrated Circuit (circuito integrato). Un chip al silicio contenente un circuito elettrico composto da transistori, diodi, resistori e condensatori. I circuiti integrati sono più piccoli, più veloci e più efficienti dei circuiti individuali utilizzati nei vecchi computer.

Incremento: Aumento di un valore specifico di una variabile indice o di un contatore.

Indice: Contatore variabile all'interno di un loop FOR...NEXT.

Indice: Variabile o costante che si riferisce ad un elemento specifico di una matrice per mezzo della sua posizione all'interno della matrice stessa.

Indirizzo: Etichetta o numero che identifica il registro o la locazione di memoria in cui viene memorizzata un'unità di informazione.

Individuazione di collisione: Individuazione di una collisione di sprite con altri sprite o con dati dello schermo.

Individuazione di errori: Routine software che identificano e spesso correggono dati introdotti in modo errato.

Input: Dati introdotti nel computer per l'elaborazione. Le sorgenti di input comprendono tastiera, disk drive, Datassette o modem.

Interfaccia: Punto di incontro tra un computer ed un dispositivo esterno, ad esempio un operatore, un dispositivo periferico o un mezzo di comunicazione. L'interfaccia può essere fisica utilizzando un connettore, o logica utilizzando un programma.

Intero: Numero intero (cioè un numero che non contiene parti frazionali), come 0,1,2 ecc.

I/O: Input/Output. Si riferisce al processo di introduzione dati nel computer, o di trasferimento dei dati dal computer al disk drive, alla stampante o a un dispositivo di memorizzazione.

Istruzione: Istruzione BASIC contenuta in una riga di programma.

Kilobyte (K): 1024 byte.

Linea commutata: Linea telefonica normale che può essere utilizzata come mezzo di trasmissione per le comunicazioni di dati.

Linea dedicata o linea affittata: Collegamento di una linea telefonica speciale, fornito dalla SIP richiesto da alcuni computer o terminali dove il collegamento è sempre aperto (linea affittata esclusiva).

Linguaggio macchina: Linguaggio comprensibile dal computer. Il computer converte tutti i linguaggi evoluti, come ad esempio il BASIC, in linguaggio macchina prima dell'esecuzione di qualsiasi istruzione. Il linguaggio macchina è scritto in forma binaria in modo che il computer possa eseguire le istruzioni direttamente. Chiamato anche codice macchina o codice oggetto.

Locazione di memoria: Indirizzo specifico di memorizzazione nel computer. Nel Commodore 128 sono presenti 131.072 locazioni di memoria (da 0 a 131.071).

Loop: Parte di programma eseguita ripetitivamente per un dato numero di volte.

Loop di ritardo: Loop vuoto FOR...NEXT che rallenta l'esecuzione di un programma.

Matrice: Struttura di memorizzazione dati in cui una serie di costanti o variabili correlate viene memorizzata in locazioni di memoria consecutive. Ogni costante o variabile contenuta in una matrice viene chiamata elemento. All'elemento si ha accesso tramite un indice. Vedere Indice.

Memoria: Locazioni di memoria all'interno del computer. ROM e RAM sono due diversi tipi di memoria.

Memoria a bolle magnetiche: Tipo relativamente nuovo di memoria che utilizza minuscole "tasche" magnetiche o "bolle" per la memorizzazione dei dati.

Memoria di colore: Area di memorizzazione del Commodore 128 che controlla il colore di ogni locazione di memoria dello schermo.

Messa a punto: Correzione degli errori in un programma.

Microprocessore: CPU contenuta in un singolo circuito integrato (IC). Tra i microprocessori utilizzati sui personal computer Commodore vi sono il 6510 e lo Z80.

Modalità: Stato operativo.

Modalità a matrice di punti: Modalità grafica avanzata del Commodore 128 con cui è possibile controllare ogni punto dello schermo.

Modalità a matrice di punti multicolor: Modalità grafica che permette la visualizzazione di uno dei quattro colori per ogni pixel all'interno di una griglia di 8 x 8 caratteri. Vedere Pixel.

Modalità burst: Modalità speciale ad alta velocità per la comunicazione tra disk drive e computer, con la quale le informazioni vengono trasmesse ad una velocità elevata.

Modalità carattere multicolore: Modalità grafica che permette la visualizzazione di quattro colori all'interno di una griglia di 8x8 caratteri.

Modalità carattere standard: Modalità di funzionamento su cui è impostato il Commodore 128 all'accensione o in fase di scrittura programmi.

Modalità diretta: Modalità operativa che esegue i comandi BASIC subito dopo aver premuto il tasto RETURN. Viene chiamata anche Modalità immediata. Vedere Comando.

Modalità full-duplex: Permette a due computer collegati alla stessa linea di trasmettere e ricevere dati contemporaneamente.

Modalità half-duplex: Permette la trasmissione in una sola direzione per volta; mentre un dispositivo sta trasmettendo, l'altro dispositivo può solo ricevere i dati e, al termine, potrà trasmettere.

Modem: Acronimo di MOdulatore/DEModulatore. Dispositivo che trasforma i segnali digitali del computer in impulsi elettrici per la trasmissione via linea telefonica e viceversa per la ricezione.

Modem a collegamento diretto: Modem digitale non acustico.

Monitor: Dispositivo di visualizzazione che ha l'aspetto di un televisore, ma che possiede una più alta risoluzione.

Monitor composito: Dispositivo utilizzato per ottenere la visualizzazione a 40 colonne.

Monitor RGBI: Intensità/Rosso/Verde/Blu. Dispositivo di visualizzazione ad alta risoluzione per ridurre il formato a 80 colonne.

Numero casuale: Numero con nove posizioni decimali da 0,000000001 a 0,999999999 generato dalla funzione RND.

Operatore: Simbolo che informa il computer di eseguire un'operazione matematica, logica o relazionale sulle variabili, costanti, o elementi di matrice nell'espressione. Gli operatori matematici sono +, –, *, /, e ↑. Gli operatori relazionali sono <, =, >, <=, >= e <>. Gli operatori logici sono AND, OR, NOT, e XOR.

Ottava: serie completa di otto note nella scala musicale.

Parola: Numero di bit trattati dalla CPU come se fossero un'unità. In una macchina a 8 bit la lunghezza della parola è 8 bit, in una macchina a 16 bit, la lunghezza della parola è di 16 bit.

Passo: Il passo di una nota è determinato dalla frequenza della onda sonora. Più la frequenza di una nota sarà alta, più alto sarà il passo. Vedere Frequenza.

Periferica: Dispositivo accessorio collegato al computer (es. disk drive, stampante, modem o joystick).

Piastra madre: In un sistema bus la piastra che contiene le linee del bus ed i connettori di testa per la sistemazione delle altre piaste del sistema.

Pixel: Termine informatico per indicare l'elemento di un'immagine. Viene chiamato pixel ogni punto dello schermo che forma una immagine. Ogni carattere viene disposto sullo schermo all'interno di una griglia formata da 8x8 pixel. Lo schermo completo è composto da una griglia di 320x200 pixel. In modalità a matrice di punti ogni pixel corrisponde ad un bit nella memoria del computer.

Polling: Metodo di controllo delle trasmissioni utilizzato da alcuni sistemi computer/terminale, nei quali una stazione "master" richiede a turno ai molti dispositivi collegati ad un mezzo di trasmissione comune se devono trasmettere informazioni.

Porta: Canale attraverso il quale i dati vengono trasferiti verso e dalla CPU. Una CPU a 8 bit può indirizzare 256 porte.

Porta parallela: Porta utilizzata per la trasmissione simultanea dei dati un byte alla volta sui cavi multipli (un bit per cavo).

Porta seriale: Porta utilizzata per la trasmissione seriale dei dati; i bit vengono trasmessi uno dopo l'altro attraverso un cavo singolo.

Posta elettronica: Servizio di comunicazione per utenti con il quale i messaggi vengono inviati ad un computer centrale, da dove il destinatario potrà richiamarli.

Priorità delle operazioni: Sequenza in cui vengono eseguiti i calcoli nelle espressioni matematiche. Chiamata anche Gerarchia delle operazioni.

Programma: Serie di istruzioni che ordinano al computer di eseguire un'operazione specifica. I programmi possono essere memorizzati su dischetto o cassetta, risiedere nella memoria del computer o essere listati su una stampante.

Programmabile: Che può essere elaborato per mezzo delle istruzioni del computer.

PROM: Acronimo di Programmable Read Only Memory (Memoria programmabile di sola lettura). Memoria a semiconduttore il cui contenuto non può essere modificato.

Protocollo: Regole seguite per lo scambio di informazioni tra il computer, compresa l'organizzazione delle unità di dati da trasferire.

Puntatore: Registro utilizzato per indicare l'indirizzo di una locazione di memoria.

RAM: Random Access Memory (Memoria ad accesso casuale). Area programmabile della memoria del computer da cui è possibile leggere e su cui è possibile scrivere (eseguendo modifiche). Tutte le locazioni RAM sono accessibili in ogni momento ed in qualsiasi ordine. Il contenuto della RAM viene cancellato allo spegnimento del computer.

Registro: Qualsiasi locazione di memoria nella RAM. Ogni registro memorizza un byte. Un registro può memorizzare qualsiasi valore da 0 a 255 in forma binaria.

REM (note): Commenti utilizzati per documentare un programma. Le note non vengono eseguite dal computer, ma vengono visualizzate nel listato del programma.

Rete ad accesso multiplo: Sistema flessibile con il quale ogni stazione può accedere in qualsiasi momento alla rete; sono previsti accorgimenti per la trasmissione contemporanea di due computer.

Rete ad anello: Sistema in cui tutte le stazioni sono collegate formando un loop continuo o cerchio.

Rete locale: Uno dei sistemi di trasmissione dati a breve distanza caratterizzato dall'uso comune di un mezzo di comunicazione da parte di molti dispositivi e dall'alta velocità di trasmissione. Chiamata anche LAN (Local Area Network).

Riga di programma: Istruzione o serie di istruzioni precedute da un numero di riga in un programma. La lunghezza massima di una riga di programma sul Commodore 128 è di 160 caratteri.

Rilascio: Velocità alla quale il volume di una nota musicale decresce dal livello di sostegno fino a zero.

Risoluzione: Densità di pixel sullo schermo che determina la precisione dei particolari dell'immagine visualizzata.

ROM: Read Only Memory (Memoria a sola lettura). Parte permanente della memoria del computer. Il contenuto delle locazioni ROM può essere letto ma non modificato. La ROM del Commodore 128 contiene l'interprete di linguaggio BASIC, le configurazioni dell'immagine del carattere e parti del sistema operativo.

RS-232: Standard raccomandato per specifiche elettroniche e meccaniche di porte di trasmissione seriali. La porta di utente parallela del Commodore 128 può essere utilizzata come porta seriale se si accede ad essa tramite un programma, a volte aggiungendo un dispositivo di interfaccia.

Salto: Saltare ad un dato punto del programma ed eseguirlo. Esempi di questo tipo di istruzioni sono GOTO e GOSUB.

Schermo: Unità di visualizzazione che può essere un apparecchio televisivo oppure un monitor.

Schermo, codice di: Numero assegnato per rappresentare un carattere nella memoria di schermo. Battendo un tasto della tastiera, il codice di schermo del carattere battuto viene introdotto automaticamente nella memoria di schermo. È inoltre possibile visualizzare un carattere memorizzando il codice di schermo relativo direttamente nella memoria di schermo per mezzo di un comando POKE.

Schermo, memoria di: Area della memoria del Commodore 128 che contiene le informazioni visualizzate sullo schermo.

Sintassi: Regole grammaticali di un linguaggio di programmazione.

Sistema operativo: Programma incorporato che controlla ogni azione intrapresa dal computer.

Sistema operativo su disco; Programma utilizzanto per trasferire le informazioni da e su un disco. Viene spesso chiamato DOS.

Software: Programmi (serie di istruzioni) per il computer memorizzati su disco, nastro o cartuccia che possono essere caricati nella memoria ad accesso casuale. In pratica il software indica al computer le azioni da intraprendere.

Sostegno: Volume medio di una nota musicale.

Sprite: Immagine grafica ad alta risoluzione, programmabile e animata. Viene chiamata anche MOB (Movable Object Block).

Stampante: Dispositivo periferico che effettua la stampa su un foglio di carta (hard copy).

Stringa: Carattere o serie di caratteri alfanumerici preceduti e seguiti da punti interrogativi.

Stringa nulla: Carattere vuoto (" "). Carattere a cui non è stato ancora assegnato un codice di stringa carattere.

Subroutine: Parte di programma indipendente separata dal programma principale che esegue una funzione specifica. Le subroutine vengono richiamate dal programma principale per mezzo dell'istruzione GOSUB e devono terminare con un'istruzione RETURN.

Tastiera: Componente di input di un sistema.

Tasti funzione: I quattro tasti situati a destra sulla tastiera del Commodore 128. Ogni tasto può essere programmato per poter eseguire una serie di istruzioni. Per mezzo del tasto SHIFT è possibile ottenere otto istruzioni diverse.

Temporizzazione: Tecnica utilizzata per sincronizzare un dispositivo di invio e ricezione di dati di comunicazione modulato per la codifica delle informazioni binarie.

Tono: Suono udibile di un passo e una forma d'onda specifici.

Trasmissione asincrona: Schema in cui i caratteri dei dati vengono inviati ad intervalli di tempo casuali. Il limite per le trasmissioni tramite linee telefoniche è circa 2.400 baud (bps). Vedere Trasmissione sincrona.

Trasmissione seriale: Invio di bit di dati sequenziali.

Trasmissione sincrona: Comunicazioni di dati per mezzo di un segnale di sincronizzazione o di clock tra il dispositivo di trasmissione e quello di ricezione.

Trasparente: Descrive un'operazione sul computer che non richiede intervento da parte dell'utente.

Variabile: Unità di memorizzazione che rappresenta una stringa o un valore numerico modificabile. I nomi delle variabili possono essere composti da un numero illimitato di caratteri, ma il Commodore 128 memorizzerà solo i primi due. Il primo carattere deve essere una lettera.

Velocità dei dati o velocità di trasferimento dati: Velocità alla quale i dati ventono inviati ad un computer di ricezione - espressa in baud, o bit per secondo (bps).

Voce: Componente per la produzione sonora all'interno del chip SID. Con questo chip sono possibili tre voci. In questo modo il Commodore 128 può produrre simultaneamente tre suoni diversi. Ogni voce consiste di un generatore di tono oscillatorio/forma d'onda, di un generatore di inviluppo e di un modulatore d'ampiezza.